Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelor Thesis

Force Sensing Book

Enrico Dominik Milutzki

4milutz@informatik.uni-hamburg.de
Studiengang Informatik
Matr.-Nr.: 6671784

First Advisor: Dr. N. Hendrich
Second Advisor: Y. Jonetzko

Abgabe: 11.2021

"Simplicity does not precede complexity, but follows it." – *Alan J. Perlis*

Contents

List of Figures

1 Introduction

1.1 Overview

Robotics is a broad topic in research as well as in industry. They are already being used for a wide variety of applications, such as electronics manufacturing, operations in the metal industry [OH21] or car manufacturing [car20]. Car parts are welded together on the assembly line by robotic arms at record speed and with the utmost precision. They are an integral part of manufacturing processes. They pave the way to Industry 4.0 [Kha19]. One of the biggest goals of robotics will always be to get closer and closer to the humans. The realization of the hand and the fingers still poses various problems. In particular, the force and tactile perception of the fingertips, which are necessary for countless actions - such as reaching for an object - are extremely difficult to model and have still not been ultimately solved: The measurement and modeling of these said forces during the manipulation of objects remains an important open research question. So far, the forces are mostly measured with sensors attached to the fingers. While these are good at measuring the forces exerted, they have the disadvantage of limiting and consequently attenuating the tactile perception. A good example of this phenomenon is, for example, the attempt to grab with gloves the appropriate key from the pants in winter. The glove prevents us from feeling the touches sharply enough and thus makes it difficult for us to localize the touches. This impairment can of course be reduced by reducing the thickness of the glove, but it may never be completely eliminated.
Another, well executable and promising possibility to measure said forces without compromising tactile perception would be to place the sensors not on the finger, but in the manipulated object. Thus, the human movements would be unhindered and the sensors could record incoming forces without the described disturbing factors. In this bachelor thesis an object in the form of a book described in exactly this way is to be designed. The goal of this book is to record a variety of common manipulation tasks that can additionally be subsequently analyzed. The majority of the bachelor thesis revolves around the design and construction of the book and the associated software. An attempt is made to answer the question whether more accurate measurement results can be achieved with sensors in the manipulated object than with sensors in the hand of the manipulator.

1.2 Contributions

In connection with this bachelor thesis, a sensory object in the form of a book was designed. The task of the book is to record certain manipulation tasks. The manipulation tasks include the following natural actions:

- Pushing a book across a table.
- Pulling a book from a shelf between other books.
- Picking up a book from a table.
- Re-grasping a book with both hands.
- Opening the book and turning a page.

Except for the last manipulation task - capturing the forces when opening the book and the first page - all originally intended tasks can be successfully captured. The difficulties will be explained in more detail in a later chapter. The book-like object should consist of a small compact core with electronics and an outer shell - disguised as a book. The core is already in the size of a trade-ready reference book, so that it is already functional as a senor book. The first idea was to create different outer shells. The idea behind the shells was to create them in different sizes and weights to simulate different books. This would have allowed a wider range of data to be obtained. However, this was beyond the scope of this bachelor thesis. But the ideas and problems will be discussed in a later section. For the motion detection of the object and the measurements of incoming forces, the book consists of several sensors. The hardware of the book is listed here in more detail:

- a WiFi-Microcontroller Arduino Nano IOT. It also contains the used Inertial Measurement Unit (IMU) to measure acceleration and the angular velocity.
- five strain gauges with HX711 modules for measuring the applied finger forces.
- a powerbank

The hardware is built into the book. It can be addressed and configured through the WiFi connection. In addition to the book, the bachelor thesis also includes the required software. This includes the code executed on the Arduino Nano as well as the host computer software for data acceptance as well as analysis. The book measures incoming forces when switched on and sends data to the host computer via WiFi. This controls the book and analyzes data. Manipulation tasks are performed with the book. With the result of the data collected and problems in the course of the bachelor thesis, the core question is tried to be answered last but not least. The Code and SCAD-Files can be found at "https://github.com/enricohs/bachelor_thesis"

2 Design

The construction, the design and the problems encountered are highlighted in this chapter. In the first design steps, the sensors are considered. Then a 3D-model is created that represents the sensory book. With the help of this 3D-model, the housing is printed with a 3D-printer and then completed with the installation of the hardware.

2.1 Chosen Hardware

First of all, it must be clarified which hardware and sensors are to be used. These must be installed ideally and without functional interference. Thus, the sensors have a large and significant influence in the design of the object. The book should be able to measure incoming forces and capture them. It should detect when it is moved - for example, along the table or when it is to be picked up. It would be advantageous if it could measure its current direction and speed. an inertial measurement unit (IMU) is an ideal sensor for the latter point. An IMU is a spatial combination of several inertial sensors such as accelerometers and angular rate sensors. Fortunately, there is a microcontroller from Arduino, which already has an IMU integrated: The Arduino Nano 33 IoT. The Arduino Nano is built up as shown in figure 2.1.

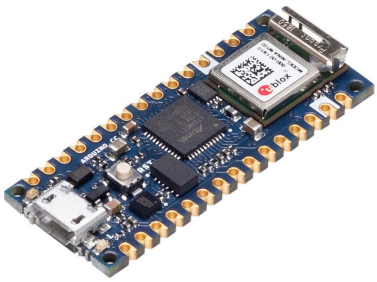

Figure 2.1: Arduino Nano 33 IOT [nan21a].

The IMU on the Arduino Nano has six degrees of freedom and meets our requirements. The six degrees of freedom come from the accelerometer and the gyroscope. The accelerometer is an electronic device on the board that can measure acceleration forces. It can orient itself using the static force of gravity, which can be measured with the gyroscope. The accelerometer measures the acceleration in one axis, the gyroscope measures the rotational acceleration around one of the axes - in each case the three axes are measured simultaneously [acc21b] [acc21a]. This allows the relative position as well as the

tilt of the IMU and thus the book to be recorded. In addition, the direction in which the book moves should be recognizable. The Arduino Nano also has the advantage of being one of Arduino's smallest microcontrollers with $45mm \cdot 18mm$ [nan21b]. Thus, it is small, light and therefore easy to install. In addition, it is also cheaper than the larger alternatives. However, the disadvantage is the few pins in the microcontroller that are needed for connecting additional sensors or modules. However, the number of digital pins of 21 should be sufficient for the book. Alternatively, a larger model would have to be selected and, if necessary, an extra IMU-module would have to be added. But as it should turn out at the end of the sensor selection and design, the case does not occur and the Arduino Nano 33 IoT can be used for the book. The Arduino Nano also has a WiFi Module. "The NINA-W102 module is a standalone multi-radio MCU module that integratesa powerful microcontroller (MCU) and a radio for wireless communication. The radiosupports Wi-Fi 802.11 b/g/n in the 2.4 Hz ISM band and Bluetooth® v4.2 (Bluetooth®BR/EDR and Bluetooth® low energy) communication" [ard21a]. Accordingly, the Arduino Nano 33 IoT also allows you to avoid buying an external WiFi-capable component. This saves space and costs. By and large, the versatility of the Arduino Nano facilitates the design of the book. The module includes control of the book, communication with the host PC and also includes an IMU.
All other sensors are operated and supplied with power by the microcontroller. In the technical data of the Arduino Nano you can find that the Arduino needs an operating voltage of 5V [nan21b]. Since the book must be freely movable, the book cannot be supplied with stationary power. It has to be powered by a battery. A power bank is the great solution here, as they are built for mobile use and are classified as a secondary battery cell - They are rechargeable. With the help of a power bank, the book can be used separately from an external stationary power supply for a longer period of time and then recharged. As with the choice of microcontroller, weight, size as well as cost must be taken into account when making the decision. The Mobile Power Bank LOGILINK PA0207 with 3000mAh and an output of 5V was chosen [pb-21].

Figure 2.2: Power Bank LOGILINK PA0207 [pb-21].

The said power bank shown in figure 2.2 has a reasonable size with the dimensions of $86mm \cdot 61mm \cdot 13.8mm$, which is easy to install inside the book. The Arduino Nano itself only needs $19mA$ in normal operation [nan21b]. A capacity of $3000mAh$ is more than

sufficient, as it provides a long operation time and the Book only needs to work for tens of minutes.
With the microcontroller and power supply successfully selected, the essential components for accomplishing the tasks must now be chosen: The force sensors. It must be remembered that the form of the book should be preserved. If possible, the sensors should remain hidden inside the book. To equip all pages of the book with force sensors would spatially clutter the interior. The manipulation tasks in which you pick up the book, carry it, or hold the book with both hands all involve actions in which you touch and hold the book on the front or back. Therefore, the idea was to record force effects on exactly these surfaces. In addition, measurements are to be taken when a book is pulled from the shelf. For this, a person usually presses on top of the standing book to tilt it. This makes it easier to grab the book from the shelf. To do this, a sensor must measure the weight acting on the book when it is upright.
For all these surfaces the same sensor was chosen, a load cell which measures through strain gauges. Strain gauges measure stretching and compressing deformations. This works by changing the internal electrical resistance even at the smallest deformations. The greater the deformation, the greater the change in electrical resistance.

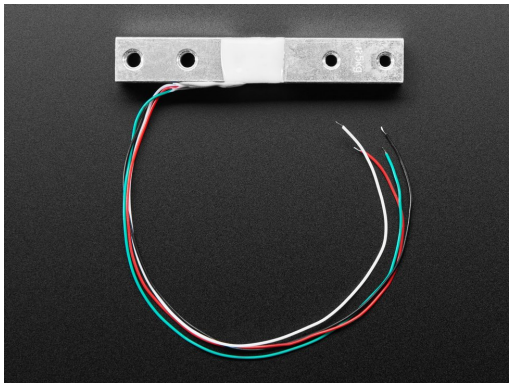

Figure 2.3: Strain-Gauge, sideways [loa21]

The load cell is constructed as shown in figure 2.3. The figure 2.3 shows the sensor lying on its side. The load cells were already on hand and their exact dimensions of $80mm \cdot 17.5mm \cdot 17.5mm$ were measured. The strain gauge located in the center of the load cell measures deformations between the two sides of the sensor. It mainly detects deformations along the Y-axis - the same orientation as the drills [str21]. For example, if the load cell is attached to the floor with the left side and supports a surface with the right side, it can measure the applied force in weight on the load cell. It is then only necessary to calibrate the cell in order to convert the values obtained into grams. In this way, it is also possible to set up a scale, as was also put into practice in the paper [**?**]. By means of the load cell cables leading out of the strain gauge - which is hidden under a white protective foil - the sensor can be operated and the measured values read.
An additional hardware device is required for this. The HX711 A/D converter module

shown in figure 2.4 is an ideal choice for this. The HX711 has already been used many times for an Arduino and also finds its use in the balance of the paper [?]. The module converts the analog signals of the strain gauge into digital signals and then passes them on. So the communication between sensor and microcontroller can be ensured.

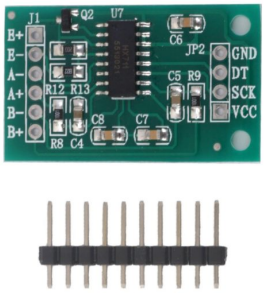

Figure 2.4: a HX711 24 bit amplifier module

The HX711 module is available in from many suppliers, but always serves the same purpose. In the end, a cheaper already existing HX711 module was chosen for the design than the one from the picture 2.4. But the book can work with any HX711 module of similar size. The used model has the measured dimensions of $33.6mm \cdot 20.5mm$. Using all these selected hardware components, the book can capture incoming forces. Movements of the book as well as the force applied when picking up the book should be well processed and sent.

2.2 Sensor preparation

The HX711 module still needs to be adapted for our purposes. The adjustment can be seen in picture 2.5. This was also done on the HX711 module of the balance from the paper [?]. Timing problems may occur during transmission to the HX711. Therefore, the weight is not measured correctly and regular outliers or repeated measurements are the result. A remedy for this is to lift pin 15 on the HX711 and connect it to VCC. This way the conversion runs more often and constant. A consistancy in the values can be achieved.

This change is expected to improve the resolution of the sensor. Due to a desired high frequency in which the data of the sensors are addressed, such an adaptation is very desirable. For this change the actual connection on the module from pin 15 to ground is removed. Afterwards it is checked with a device, that the connection to Grounded is really removed, so that the current can find its way to Power supply voltage "VCC" (Voltage at the common collector) during operation. afterwards a small resistor is soldered to the VCC connection for safety. A cable then connects VCC to the desired connection - pin 15 - of the chip. It is checked with a meter every now and then that the resistor is not too large and that the current can easily pass through. With these changes and the

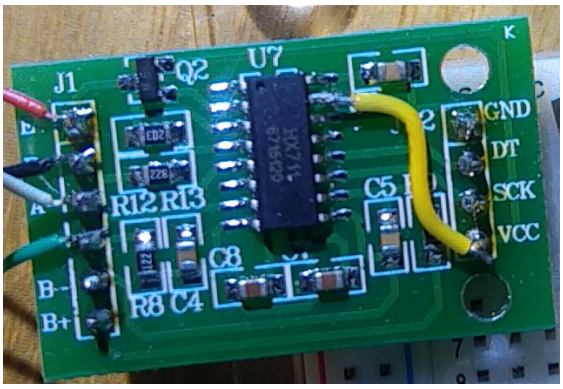

Figure 2.5: HX711-Module adjustments EIGENES BILD EINFÜGEN

precautions, the chip is finally prepared for operation inside the book.

2.3 Hardware Pinout and Spreedsheet

The wiring diagram of the book sees the Arduino Nano as the central component. It runs software, drives sensors, reads measurement data and sends it to the host. Powered by the power bank, a micro USB connection runs from the power bank port to the Arduino Nano micro USB port. Additionally the HX711 modules have to be wired for communication with the Arduino. Furthermore, the HX711 and the associated load cells are supplied with power by the Arduino Nano. Each of the HX711 modules is wired to the Arduino in the same way as shown in the figure 2.6.

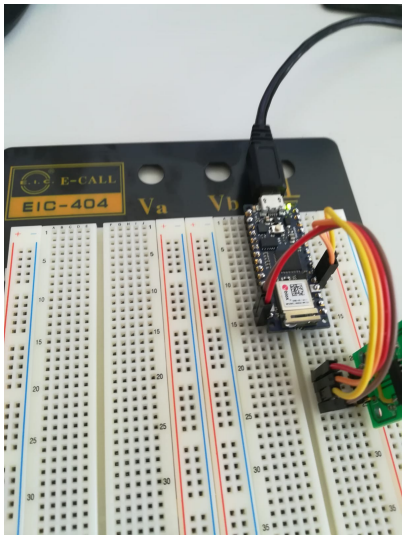

Figure 2.6: Arduino - HX711 added wiring

The grounded output of the HX711 is connected to the grounded port of the Arduino. Power is supplied to the HX711 via its VCC connector, which is connected to the 5V pins of the Arduino. Finally, however, the task of the HX711 is set to frequent and the data read

in. For this the DT connector of the HX711 is connected to a pin of the Arduino that can process digital signals and the Clock (CLGor SCK) connector of the HX711 is connected to another pin that can also process digital signals. In figure 2.6 the connections to the digital pin 2 and the digital PWN pin 3. The Arduino Nano has 14 such digital pins. The Arduino has two of each of the 5V pins and the grounded pins. But since only one of the Arduino pins 5V and Ground can be used by all HX711 modules, the only limitation are the 14 digital pins, that can be connected to the HX711 Modules [nan21b]. It is possibly to clock multiple HX711's from a single pin, but that could cause difficulties. Fortunately the digital pins of the Arduino Nano are sufficient. Since each HX711 only needs two of the 14 pins and five sensors are used in the book, only ten pins are connected. These are also the only connections of Arduino Nano that have been built in the sensory book. The connection that has not been explained yet is the connection between the HX711 amplifier and the strain gauges of the load cells.
The circuit is illustrated in figure 2.6. The HX711 controls the sensor and supplies it with power, which it receives from the Arduino Nano. Because of the white protective layer on the load cell, which isolates the strain gauge from the outside world, it is not possible to see from which connector the load cell cables lead. The correct compliance of the wiring must be assumed to be guaranteed. The HX711 ports E-, E+ as well as A- and A+ must be addressed with the sensor. These must be connected to the same designation ports of the sensor. The E+ ports are connected with the red cable, the E- ports with the black cable, the A- ports with the white cable and last but not least the A+ ports with the green cable. The last connection not yet mentioned is not a switched connection. It involves communication from the sensory book to the host computer. Through a Wifi connection between the Arduino Nano and the host PC, information can be exchanged wirelessly.

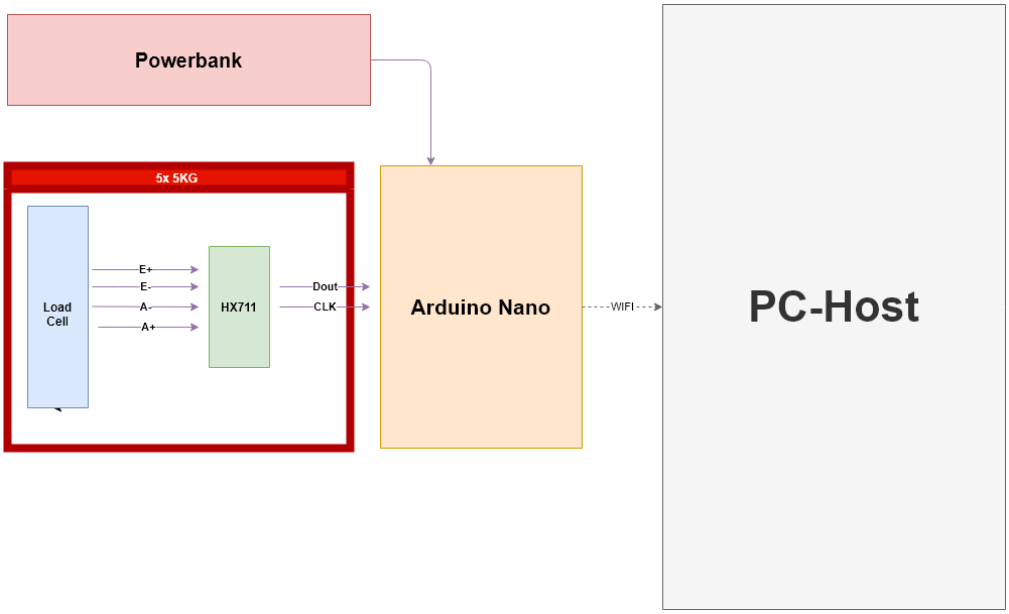

Figure 2.7: Book circuit diagram

Through all the explained connections and interconnections, the circuit diagram 2.7 for the project is created. The circuit diagram is the essential building block of a construction and ensures hardware functionality of the sensory book.

2.4 Designing the Book

In this section, the design of the book is discussed. For the modeling the program OpenSCAD was used. With this program a script is written programmatically, which generates the 3D model. The code is included with a separate file. A short code description can be found in the next section "2.5 OpenSCAD Code description".
The Design must be achieved to create an object that looks like a book and can be handled like a book. It should measure and record the force effects of a book - displacement, acceleration and impacting weights. After the first weeks of thinking about the rough shape of the sensoring object, about which manipulation tasks to handle and which sensors to consider, it must be able to think about more precise details. As already shown in the "2.1 Chosen Hardware" section, it must be possible to measure especially the incoming forces of the bottom side and the opposite top side - the book cover. Care must be taken to ensure that the only connection between these surfaces is via the load cells. The outer walls of the book interior must not transmit incoming forces from one surface to the other surface and thereby falsify the results. However, they are necessary to protect the interior from the external forces, make the object look more like a book, and make it more manageable for the required manipulation tasks. Therefore, the idea has been to attach the interior to the Book bottom. The bottom of the book has the dimensions of $190mm \cdot 150mm \cdot 2mm$ and is surrounded by 4cm high walls. The walls are each 2mm of thickness, which makes the total construct $192mm \cdot 152mm \cdot 40mm$. The interior of the book button then contains all the hardware components. The cover side of the book should then added to the design and close the Box. The cover page achieves this without touching the walls and being carried by them. The cover page of the book is connected to the bottom of the book via the sensors like a lid. With this rough draft of the external appearance, the question now arises as to how and where the force sensors are to be placed. it has already been indicated that it was not only decided for one Load Cell for the recording of the acting forces on the large surfaces of the book. First of all, it only needs one sensor to measure the incoming forces on both sides. The sensor can be calibrated in such a way that, for example, the forces acting on the cover side are represented as forces in the positive number range and the forces acting on the back side of the book are represented in the negative number range. This way half as many sensors are required.
Four Load Cells are bolted to the top and bottom surfaces of the book. It was decided to use four sensors in order to have one sensor for each side as a support for the book cover. The book cover should be stable and not bend too much. Therefore four sensors

are a good approach. Secondly, it should be tried to locate the point on which the force acts. The sensors that are closer to the point should absorb and measure more force than sensors that are further away. From this data the localization should be done on the host Computer.
The first idea was to attach each sensor to one side of the book - without contact between the sensor and the outer wall. Each sensor should be located at one corner. The connection from the cover to the sensors should be at the corners for stability and reduced bending possibilities. The dimensions of the book had to be adjusted after the first print, so that the space was not sufficient on all sides. Therefore, the sensors are now placed two at a time on the lengths of the bottom. However, they are still oriented towards the corners, so this does not have any disadvantages in terms of stability. Once again, it is important to note that the only connection between the surfaces is through the sensors. In addition, it must be ensured that it is possible to work with the hardware even after printing and assembling the book. Especially the power bank must be accessible. The power bank must be freely rechargeable. For this purpose, it was decided to design a book lid as a kind of grid, as can be seen in Figure 2.8. the true book cover page should be attached to the grid afterwards.

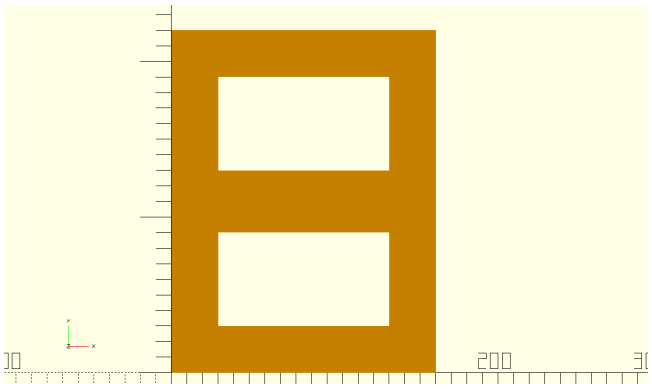

Figure 2.8: Bookcover: The Grid (OpenSCAD)

Through the two larger holes of the book lid, individual parts of the interior can be placed, taken out or otherwise processed. This allows the power bank to be placed more centrally and there is no need to create a hole for the USB charging port of the power bank. An HX711 module should not be far away from its load cell. Therefore we try to place them in a distance of one to two centimeters in the direction of the inside of the book from their respective load cell. This leaves enough space to place the central part of the construction: the Arduino Nano. In addition, there is enough space for the by far largest component: the Power Bank. Accompanying all these points, the design of the book, which can be seen in the figures 2.9 and 2.10, is created. the illustrations present a good first picture of the book.

Figure 2.9 shows the book without the grid nor cover. The HX711 modules are drawn in green, the Arduino Nano in blue, the Load Cells in grey and the power bank in white.

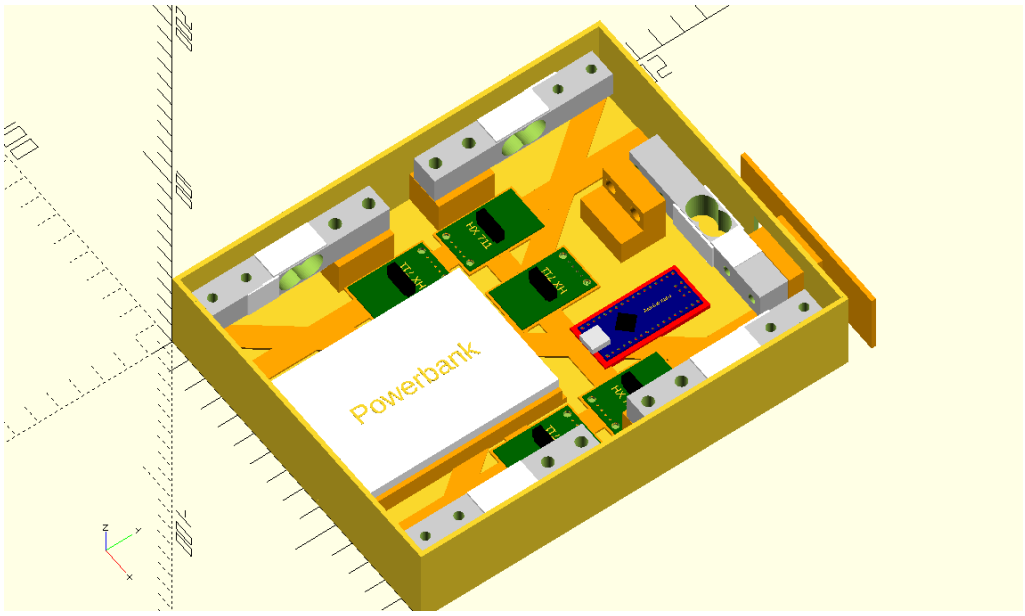

Figure 2.9: Book Design without Cover (OpenSCAD)

In the illustrations, additional elevations are shown in orange. This makes it easy to see that all hardware parts are screwed onto an additional elevation. On the one hand, this is for stability. The thin layer of the book of 2mm thickness could possibly bend in the experiment and distort the result. On the other hand, it requires space to place the screws into the material. So the elevations allow the hardware parts to be attached properly. Struts were used to support the book stability. The struts are also shown in orange. These struts help to prevent bending and are 3mm thick. They are also placed inside the book, so as not to change the surface of the object and spoil the shape of the book. In figure 2.10 you can see the whole project with the grid and cover.

The decision is to lift the grid a few millimeters from the side walls. This is to ensure that no pressure impulse is lost through the walls if the book cover should bend significantly due to increased pressure. The real book cover page is also visible here. It is attached to the grid. The dimensions of the book should be that of a standard specialist book. But the original dimensions had to be adjusted to the present dimensions due to the size limitations of the 3D printers. The illusion of a book has not been lost as a result. The book should also have a reasonable thickness. A book thickness around 4.5cm was chosen. The grid still needs to be connected to the rest via the sensors. In figure 2.11 the construction for this is visualized using a selected load cell. It must be noted, as in section "Chosen Hardware", that the same side of the Load Cell cannot be attached to the book cover as well as to the book back.

Different sides must be able to measure the stretching of the Load Cell for force detection. The sensor is lifted 17mm from the floor and attached to the support. The side of the Load Cell that is near the corner of the book is connected to the book grid. This is to prevent local bending of the grid. This design allows the incoming forces on the men-

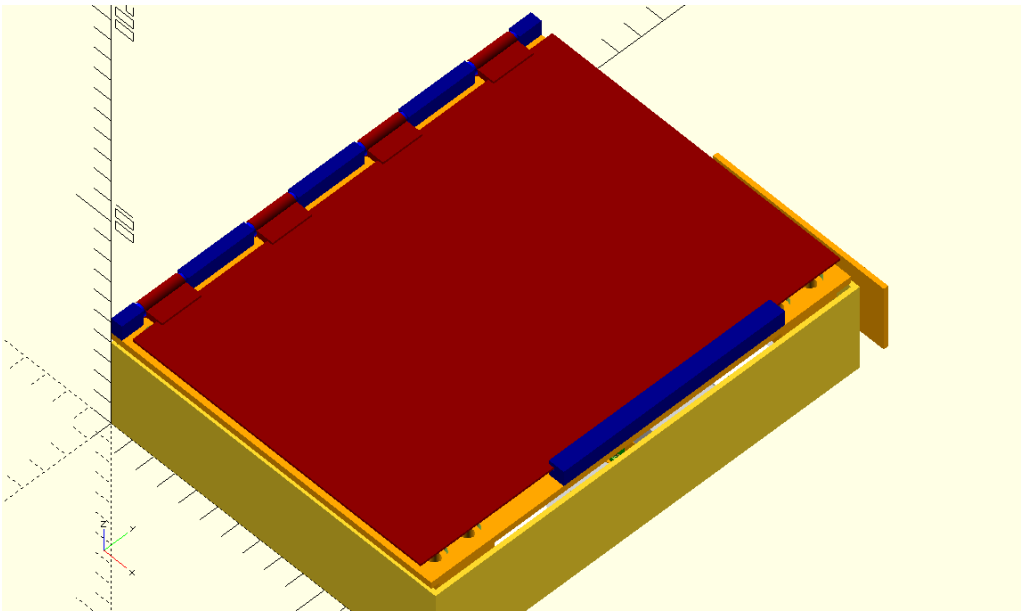

Figure 2.10: Book Design (OpenSCAD)

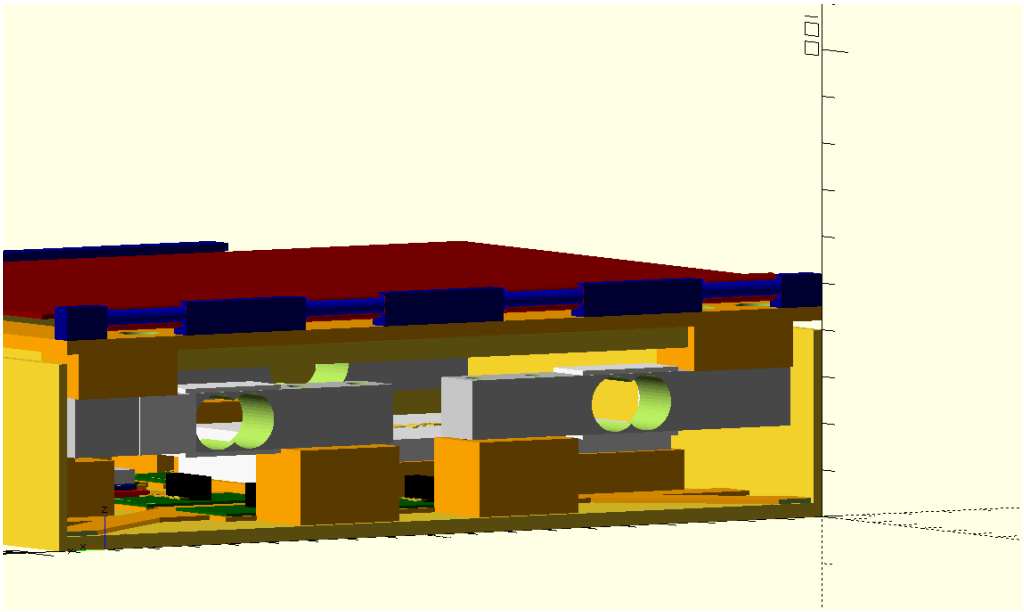

Figure 2.11: Book Load-Cell integration (OpenSCAD)

tioned surfaces to be measured and ideally allows the localization of the pressure points to be made true. The supports were first placed 1mm away from the outer wall. However, it turns out that it makes much more sense to tie the supports to the outer walls. They provide more support and the force measurement should not be disturbed - the only connection between the cover and the back of the book remains via the sensors. The sensors still stay several millimeters away from the outer walls. The holders are adjusted on the back so that the 80mm long sensors can be placed several millimeters away from the edge. So that the brackets on the book grid is also lifted at a certain distance from

the outer walls. The sensors are attached to the aforementioned holders with the help of screws. The holders have suitable holes for this purpose, which extend through the bottom of the book or through the grid. The holders have suitable holes for this purpose, which extend through the bottom of the book or through the grid. These can be seen again in Figure 2.12.

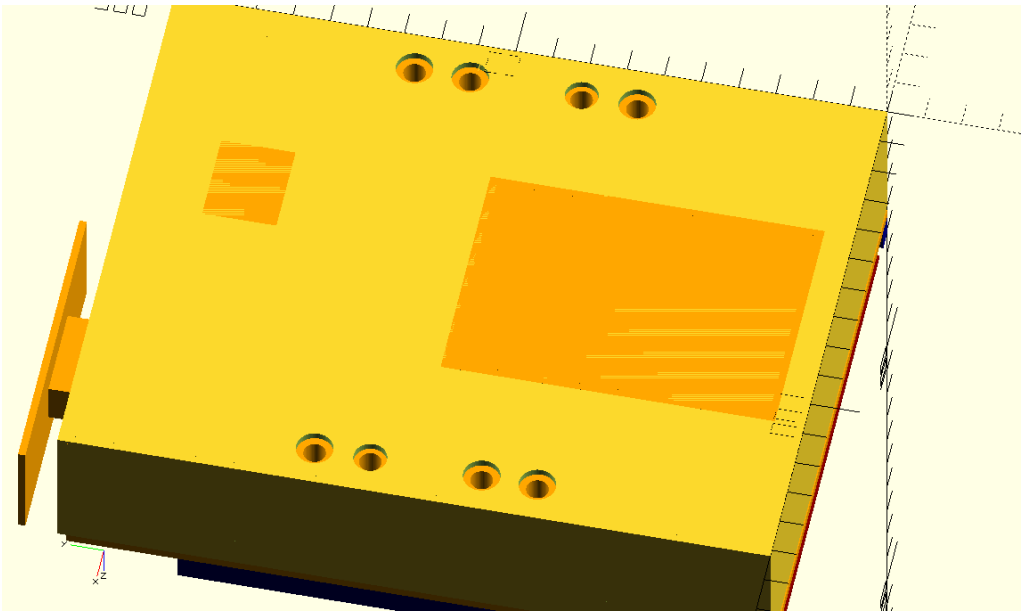

Figure 2.12: book bottom with boreholes (OpenSCAD)

The selected load cells have different hole sizes per side. M4 screws are to be installed on one side and M5 screws on the other side. For the book, all M4 screws are attached to the bottom of the book and accordingly the narrower screws to the top of the book, the grid. The idea is to drill the screws from the outside, through the plastic into the sensors. The holes of the Book must therefore be created with a radius in which the screws simply fit through. They should not mill in, but also not be pushed freely in all directions - the sensors should be fixed statically and firmly in the book. The holes on the bottom of the book have a diameter of 5.2mm and the holes on the grid have a diameter of 4.2mm. Thereby the boreholes reach through the brackets and through the respective book side. In addition, the holes in the outer walls for the screw heads are several millimeters larger. Thus, the screws do not interfere with the use and the book does not lose its book-like appearance so much.
The sensory book deals with the manipulation task of a person being able to tip the book off a shelf and then grasp it. The Book needs to detect the acceleration of the book and Weights acting on it. The placed sensor is already visible in the figures 2.9 and 2.10. The idea and functionality is again the same. With the help of a strain gauge attached to a load cell, forces applied to the surface are to be measured and recorded. Again, it must be ensured that the Load Cell lies unobstructed and is not prevented by other

components, so that the forces can be recorded unhindered. In addition, the underside of the load cell must be firmly attached to the book. The entire side, in this case an outer wall of the book, should not be selected as the surface. This would only have to be fixed by the Load Cell and lift a few millimeters off the book. This would make the construction too unwieldy, uncomfortable and look unstable. Therefore, it was decided to attach an extra smaller surface to the sensor. This will be used to tilt the book. Of course, this has the disadvantage that the book cannot be grabbed from the shelf at will. For experimental research, however, the one-sided possibility is sufficient for the time being. The implementation is shown in more detail again in Figure 2.13.

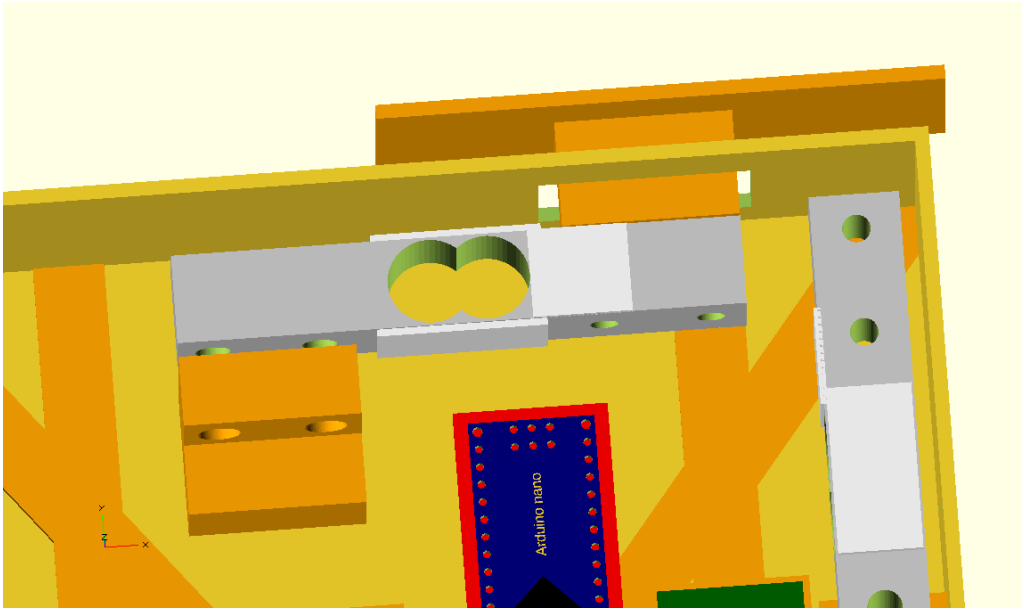

Figure 2.13: Outer Load Cell integration (OpenSCAD)

In this case the additional, smaller platform had to stand out a bit from the book. However, this flaw is to be tolerated. The Load Cell the small platform can be hold with a printed connector through a smaller hole in the wall of $20mm \cdot 30mm$. You can see the additional fixture inside the book, which should hold the construction stable to the book. First attempts were to attach the construction to the outer wall. However, it turned out that the construction protrudes too far from the book, leaving the sensor free and not isolated outside the book. In addition, it destroys the illusion of the book and interferes with the accomplishment of manipulation tasks. Here the sensor is screwed to the brackets. The diameter of the holes in the bracket is the same as the other holes on the bottom of the book. The sensor was placed at a height so that when lying down, the platform is $2mm$ in height at the center. The platform is of dimensions $80mm \cdot 30mm \cdot 2mm$. The platform is held by a perforated support, which is $25mm$ long and attached to the sensor. The platform itself has holes like those of the book grid: they are widened for the screw heads, so that the screw does not interfere with the task.

Figure 2.14 shows the layout of the book inside. Here you can see the holders of the

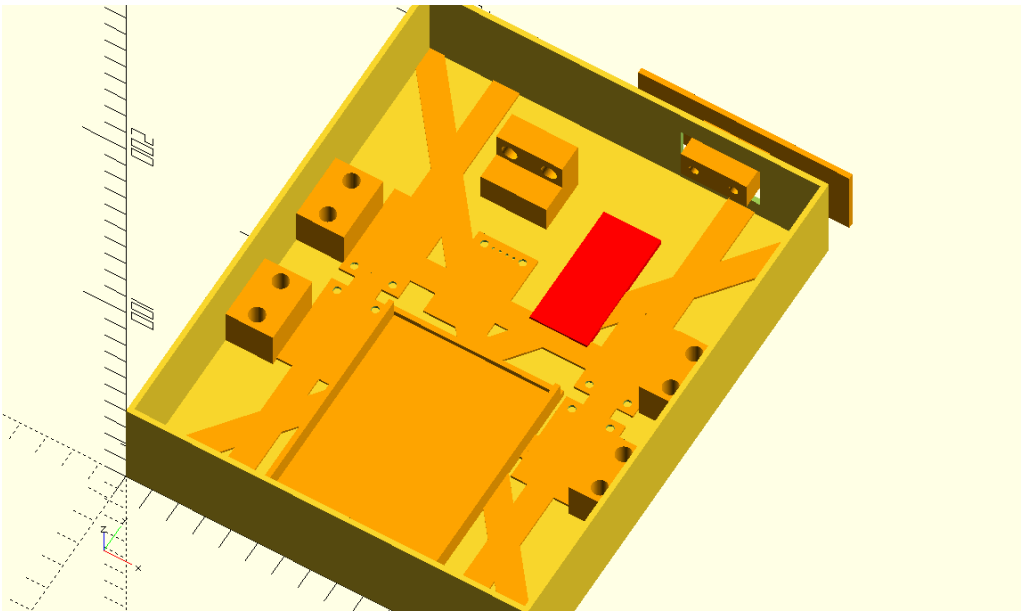

Figure 2.14: book bottom (OpenSCAD)

load cells with their holes. You can also see the elevations for the HX711 boards. These are raised by 2*mm* and give a more stable environment. This is where the HX711 modules are glued. Also all struts are 2*mm* thick. Two struts are on the bottom diagonally each from corner to corner. The Arduino Nano is glued upside down on the red marked area. In addition, there are four more straight struts. The struts are to make the floor more stable and prevent the floor from bending. Marked in red is the 3*mm* elevation for the Arduino Nano 33 IoT. The Arduino Nano is glued upside down on this surface. The Arduino Nano must be flipped over to make the pins available for all the cables. By flipping it over, the cables can be easily plugged in. Last but not least at the bottom is the power bank. The power bank has the largest bracket which is 2*mm* thick. The bracket also contains outer walls to better adjust and glue the power bank. The bracket wall in the direction to the Arduino is way lower to leave room for the cables. The power bank is placed on the flat so that the connections lead towards the inside of the book.
The book grid represents the mounting of the actual book cover. The design can be seen in figure 2.15.

The top of the book consists of two elements. In figure 2.15 one element - the cover sheet - is marked in red. The connection points for the cover page re drawn in blue. On one side, which is in the front of figure 2.15, are the holders of the cover page. The cover is to be opened like a page. Therefore, the cover page is tied to the round, cylinder-like components of the holder. These have a diameter of 3*mm*. The cover side has hinges for this, which are pressed onto the cylinder holders and clipped. After that, the cover can be used like the cover page of a book. The blue holder in the background of 2.15 has a slot in which the cover page can be clamped when closed. This is to prevent the cover page from constantly flipping open and closed during manipulation tasks.

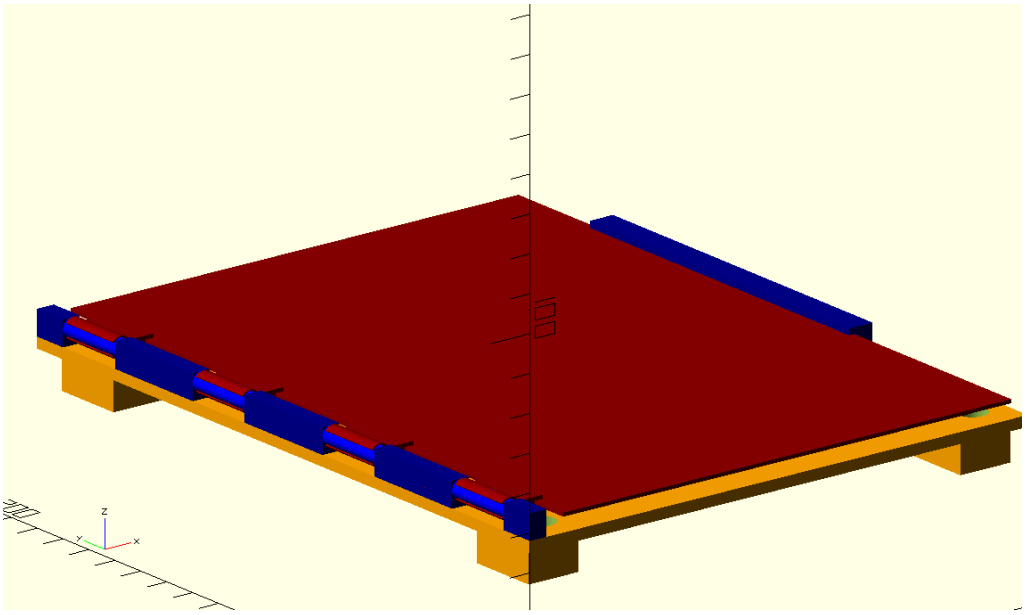

Figure 2.15: Book Cover (OpenSCAD)

2.5 OpenSCAD Code description

In this section, the code is presented in summary. In OpenSCAD, larger 3D Objects are created using simple solids. Shapes of any kind can be created by moving, rotating and combining the simple 3D objects through several functions of the language. As usual in programming or scripting languages, libraries can be used. Thus elements of the design of the scale are taken from the paper [**?**]. More precisely, the 3D models of the Load Cell and the Arduino Nano are taken over. The design of the Arduino Nano can be taken over without problems, since the dimensions between the Arduino Nano and the Arduino Nano 33 IOT correspond. The Design of the Load Cell was changed a bit to match it with the one used.
The code calls the function (named "module" in OpenSCAD) "draw_book()". In this function the book is created. Thereby the single components bottom, top, as well as hardware, cover or the hinge components for the cover can be displayed independently or not via the parameters of the function. Also the components of the tilt operation can be presented individually. In the first part of the function, in which the underside of the book is generated, first the soil with the outer walls from quarters is generated. Subsequently, the struts that provide stability to the book are created here. In addition, this part is used to set all the highlights on which the hardware parts will be attached. This is supported by auxiliary functions that build the background of the HX711 objects as well as the load cells. In the preview, the highlights are drawn in a different color than the surfaces of the book to create a clearer picture. The drill holes in the highlights for the hardware

elements have been set using cylinders. The "difference()" function can be used to pull objects away from each other to design the drill holes. This technique was also used for the diagonal struts, which are actually just cuboids. Pieces of the cuboids that would stick out of the book are removed from the model using the difference function.
In the next part of the function "draw_book()" the top of the book and the corresponding grid plate are generated. The grid is created first. The hinges are created directly. Here the "difference()" function is used to create empty spaces. Then the blue component of figure2.15 is created by the function "hinge()". Finally, the connections between the load cells and the grid are created. The same function "hx_sensor_cube()" is called for all Connection-Cubes of the Load Cells.
After that, the cover page is displayed - The page marked in red. On the cover side there are also the hinges, which are again created with the help of the "difference()" function. The hinges can then be clipped to the blue holder after printing. In the final part of the "draw_book()" function the hardware parts are generated. Here all elements are set to the given highlights one after the other. For the load cell the method "load_cell_TAL220()" created by the paper [**?**] in modified status is used. The holes had to be adjusted to resemble the actual Load Cells in use. The Power bank as well as HX711 modules are implemented by own functions. Once again, the components are identified by appropriate color selection and corresponding designations. The help functions can be found at the end of the script.
Using this code, the 3D models of the individual parts of the book cover and spine could be generated and then turned into reality by printing. The OpenSCAD scripting language proved to be an easy-to-use and efficient tool for the design of the sensory object.

2.6 3D Print

After the design the sensory book can be printed. A 3D printer needs 3D models in stl format. In the OpenSCAD program, the models must be rendered and then output as stl objects. Since the book consists of several parts, which are connected or screwed together only after printing, it needs even four 3D models:

- **sensing_book_bot.stl**: Contains the book base with the struts and all the highlighting for the hardware parts.
- **sensing_book_tilt.stl**: Includes the components for sensing the forces when tilting the standing book. This includes the connector to the sensor as well as the platform that will experience the forces.
- **sensing_book_top.stl**: This 3D model shows the grid. It includes the connectors to the sensors as well as the brackets for the cover side.
- **sensing_book_cover.stl**: In the last 3D model, the book cover page is shown.

.

These models can be displayed individually without problems using the parameters of the main function and then rendered. The 3D models are shown in the figures 2.16 and 2.17

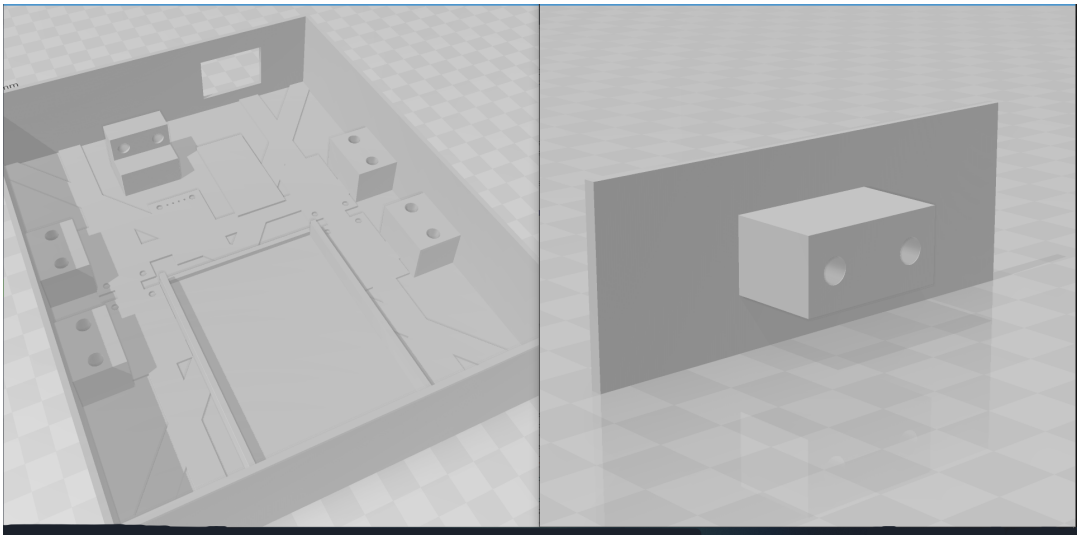

Figure 2.16: 3D Model sensing_book_bot (left) and sensing_book_tilt (right)

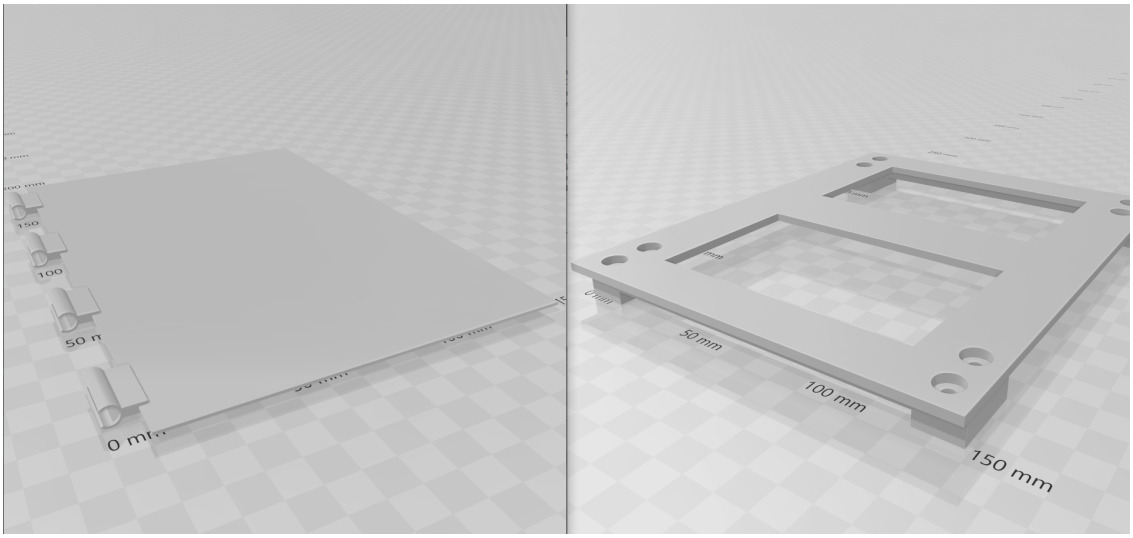

Figure 2.17: 3D Model sensing_book_cover (left) and sensing_book_top (right)

The 3D printer at the University of Hamburg uses a plastic to print the 3D models into the real world. Tor the largest model of the book, a 3D printer needs more than eight hours and is thus busy for a working day. 3D printing thus takes a relatively long time, and it is therefore not unimportant that the models are correct. However, it is not uncommon for several attempts to be needed before the desired 3D model is printed in a functional way. Thus, the sensory book was no exception. On the first attempt, the book's dimensions of $222mm \cdot 170mm$ were too large for the 3D printer and it could not

be printed properly. As a result, there were only one of two options: Either the book is printed in the dimensions in several parts, which are then glued together, or the book must be reduced in size of a maximal $200mm \cdot 200mm$. The second option was chosen and the book now measures $190mm * 170mm$. As a result, the book unfortunately no longer resembles the original dimensions of an actual reference book, but it does not lose the appearance of a book. A model of the original design can be seen in figure 2.18.

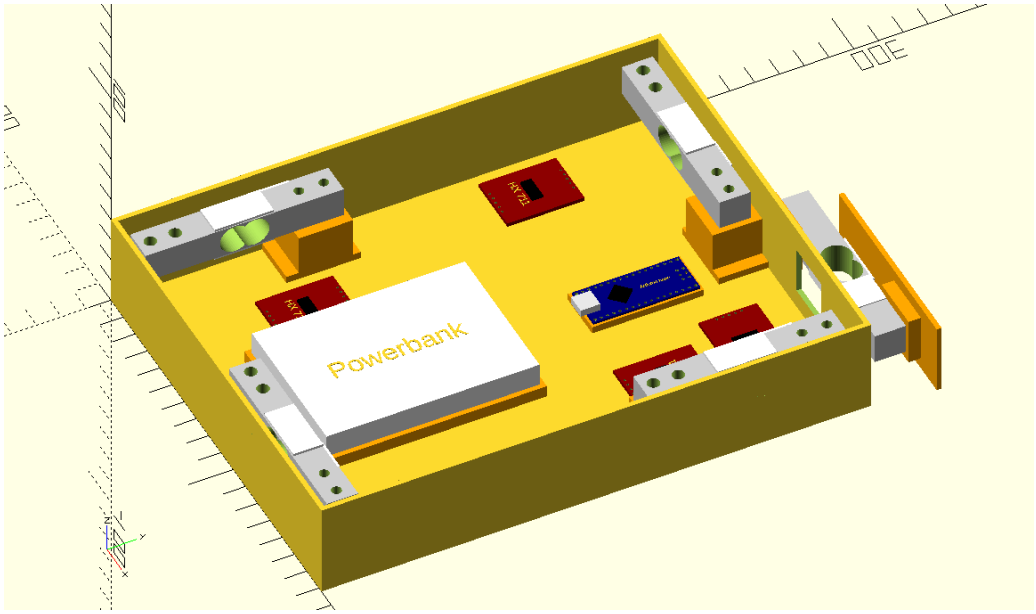

Figure 2.18: old Book bot (OpenSCAD)

It is a little bit older than the actual tried model, but the old arrangement of the sensors can be recognized here. Due to the reduced dimensions, the sensors had to be rearranged. Originally, a width of $130mm$ was aimed at to correspond to actual book dimensions again [boo21]. But this was too narrow to fit the sensors and the HX711 boards inside. The width was then increased to fit all the components. The book could be printed this way. Several deficits became noticeable. First, the two holes per side for the screws were placed at the wrong distance from each other. This meant that the load cells could not be screwed on with both screws. In addition, the holes for the screw heads were not the right diameter and could not be hidden properly. The book grid was too flexible and needed supports for stability. Then there were problems attaching the cover. The clip mechanism did not work as designed. The clippers were not pliable enough to clip to the hinges. Since the material should have a low melting point at about $50 - 70C$, an attempt was made to heat the material enough to make it more pliable. For this, the cover was tried to be heated in hot water or in the oven - Neither helped. Last but not least, the Arduino had to be turned over to get to the pins more easily. The bracket had to be removed. Despite these shortcomings, with a lot of drilling and precautions, this first model could be provisionally equipped with the sensors and screwed together for initial testing. This model can be seen in figure 2.19.

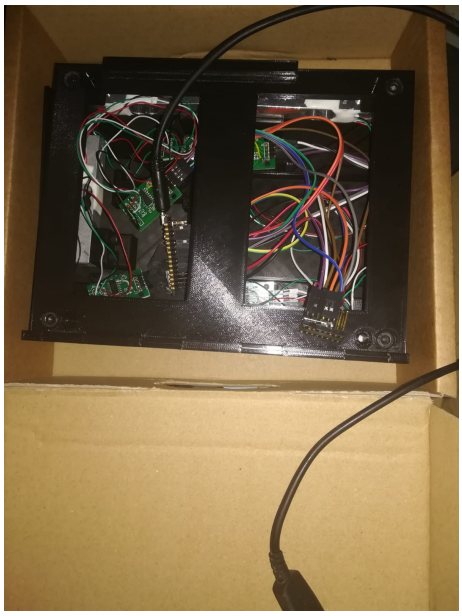

Figure 2.19: First Book

2.7 further difficulties

In this section, problems and difficulties in the course of the design are still explained. For the 3D model, the tool FreeCAD was chosen first. It has more functions than OpenSCAD, but is recognizably more tailored for engineers. In FreeCAD, one works without writing code and instead draws sketches, which are then pushed up to 3D objects. This makes working with the program very intuitive at first and you quickly get started. 2D objects can be drawn in well and by setting relationships such as a vertical distance between two objects, more complex objects can be designed. In retrospect, however, the program turned out to be more difficult than it initially seemed. It takes away freedoms that you would have in OpenSCAD. For example, it is not possible to generate several independent 3D objects without additional precautions. Because of this and many other inconveniences, I switched to OpenSCAD. As a computer scientist, programming is a familiar world and there were no limits to the advantages of free design. Getting started was easy and a big advantage was being able to use existing code modules from the paper [**?**]. This allowed the 3D models to be created more effectively and time efficiently. The already created design pieces in the previous program could be taken over conceptually.

Another and even really big difficulty was to design the originally set manipulation task of opening the book and to capture this. Even the design of the reclosable top was a problem. However, with the help of the grid, to which the sensors are attached, the problem was solved. The big problem that remains unsolved is finding the right way to detect the force of the cover being opened. Since the page is moving, no load cell can be used

for this. Also a mechanimus, in which a rope is pulled, is too disadvantageous for the purpose. This would interfere with the manipulation task. As is well known, a book has no force that pulls the cover page into the closed state. There must be an exposed page that can be opened without problems and the acting force can be measured. The book should measure the read data.
one promising avenue was the paper [HWZ20], which promises low-cost 3D printing of sensors. However, the models and ideas presented could not be adopted for the cover page problem. There is still no way to capture the forces for the side that opens up and thus leans away. No similar cases could be found.

3 Software

The book is operated wirelessly and communicates with a host computer via WLAN. This chapter describes the software for the bachelor thesis. This includes the software of the sensory object as well as the software of the host computer. Here, the Arduino Nano 33 IoT lays in the sensoring object and only performs the task of measurements. The measurement data is read in a large continuous loop and then sent to the host. However, the Arduino can also communicate via the serial console for debugging. The host is run via a Python script and takes the measurements. These are then processed and evaluated as desired. Inputs and thus commands can be entered and executed via a small input window. The host can thereby send messages also to the Arduino Nano and influence it thereby. The manipulation tasks can thus be executed in a knowledge-gaining manner and the measured values as well as findings can be stored.

3.1 Communication

The two parties communicate via the WLAN. The "User Datagram Protocol" UDP is used for this [RLM21][udp21]. The UDP is a minimal and connectionless network protocol that enables the sending of so-called datagrams. The datagrams are basically nothing more than message packages. The participants of the communication need the address as well as the chosen port of the other. A packet is sent with the data addressed to the destination. UDP is therefore a connectionless, but also non-reliable transmission protocol. It is not necessary to establish a connection before starting the transmission. This means that data can be exchanged more quickly, which is particularly beneficial for systems with messages containing small amounts of information - as is the case here in the sensory book. However, the lack of a connection also means that packets may theoretically not arrive or may arrive at their destination in the wrong order. Against these cases must be taken care, so that these do not lead to errors. The Arduino Nano also has the possibility to communicate with the Transmission Control Protocol TCP.
This protocol would be connection oriented, accordingly establishes a connection between the two parties and is used as a protocol in many data transmissions because of its advantages. TCP is used as an almost exclusive transport medium for the internet, e-mail-services and many other popular network services. Messages can be exchanged between the two parties (sockets) without losing messages. Data loss is detected and automatically corrected. The protocol has many advantages over UDP. The reason why UDP was chosen nevertheless is the time: The multiple backups of TCP require a cer-

tain latency in the message transfer. In this application, where the book constantly sends small messages with the measured values, on-time data with occasional packet loss is more valuable than slow messages with delayed retransmissions. In this system, data is sent in millisecond intervals. UDP is also an unsecured and unprotected transmission protocol. However, this plays no role in this small research project and had no influence in the choice of protocol. The guarantee that data arrives at the recipient unaltered or inaccessible to third parties is not necessary here. It is not sensitive data and the tests have to be run multiple times either way, so that outliers can be detected.
Both parties are hosts in the UDP protocol, which can receive data at any time. Therefore, the Arduino as well as the host computer run in a continuous loop, which checks in each iteration for a message and evaluates it if necessary.

3.2 Arduino Nano Software

The microcontroller has its name for a reason: it controls all the components of the system and organizes what happens. Via the Arduino IDE one has the possibility to upload software to the Arduino Nano. The Ardiuno Nano is built for embedded systems and runs the software after a setup in a continuous loop . Programming is done in the C++ programming language. The sensory book runs through the software of the Arduino. Via WiFi it is possible to send messages to the Arduino via the host PC and receive information. This chapter is dedicated to this software. It describes the idea, the basic building blocks and functionality.

3.2.1 Embedded Systems Arduino Software Basics

The Arduino software is optimized for embedded systems as already and the Arduino Nano 33 IOT is also built for it. The software running on the system is always built into two elementary blocks. At system startup, the fields are declared and the "setup()" function are executed. In the "setup()" function components are typically initialized. The setup() function is executed only once. Subsequently, the "loop" function is executed. As the name suggests, this function is executed over and over again in a continuous loop. You can think of this function in terms of a while(True) condition. This function is where the arduino's real work is done. This is where values from the sensors are read, processed and output. For feedback and debugging possibilities the serial monitor exists. It is possible that the Arduino sends messages there with any available information. In addition, Arduino offers the possibility to define commands that can be entered into the console/serial monitor. This way you can also configure the system in runtime. The software will not stop until the system is turned off.
The Arduino Nano 33 IoT in the sensory book initializes all sensors in its setup() function. In addition, a WLAN connection is established - the LED lamp of the Arduino Nano

provides information about the connection. In the loop() first the WiFi connection is checked and the LED Lamp adjusted. the values of the IMU are read and sent to the host computer. Then the Load Cell sensors are read out one after the other and the values are sent to the host computer. Last but not least it checks if a command was sent from the host computer or by the serial monitor. The Loop() always follows these steps.

3.2.2 The Software Adjustments

The Arduino Nano software from the paper [**?**] serves as a basis. For the created scale only one sensor was addressed, which was realized in the software. For the bachelor thesis five sensors and the internal IMU as well as the internal Wifi port have to be addressed. In addition, the LED lamp present on the Arduino Nano 33 IOT is to be used for feedback purposes. Arduino software is divided in its basic building blocks as follows.
The software of the scale in paper [**?**] initializes the sensor in the "setup()" function. In the "loop()" function the current value of the sensor is read and afterwards the data is output to the serial monitor.

The iteration step, the elapsed time in milliseconds, the value of the sensor, the offset of the sensor and last but not least the calibrated value is output. The offset means the currently calculated standard value, which the sensor reads if it does not detect at any weight. In addition, commands have been defined for the scale, which can be entered into the console or serial monitor. At the end of the "Loop()" function, the input is read and evaluated for this purpose. Several functions are defined, for example to output the average, to calibrate the sensor or to store the calibration values into the EPROM. The Erasable Programmable Read Only Memory (short EPROM) is a small memory unit on a Arduino to store values even after operation. This basic structure and idea will be kept for the software of the force sensing book. First, the program must be changed so that the five sensors can be operated simultaneously. This results in several adjustments to the program. First, a class "hx711_sensor" is defined, in which all variables and values of a sensor are declared, stored and managed.

The class can be seen in figure 3.2. First, an HX711 instance is declared. Using the HX711 class from the external library, the values can be read and spoken to the component. Next, all other variables are initialized with a default value. Finally the class method "$init(intdout, intclk)$ is defined. This is executed in the setup function to initialize the sensor. The pins to which the sensor or the HX711 component is connected are passed. Afterwards all functionalities, which do not count to the actual measuring, were taken from the responsibility of the Arduino. So calibration, offset determination or average determination are no longer part of the Arduino code. In the figure 3.2 the variables are still included to understand the relation to the original code. But in the final code they are removed.
In the fields of the script a list "sensor_list" of "hx711_sensor" objects is declared. In the setup, five hx711-sensor instances are added to the list and initialized with the respective

```
// Klasse mit allen Infos für ein Sensor
class hx711_sensor {
  public:
    String name_id;
    HX711 scale;

    long  scale_offset = 604949;
    float scale_gain = 1.234567;

    long tara_temp = 0;
    int  tara_in_progress = 0;
    long data[ DATASIZE ];

    long value = 0;

    long av_count = 0;
    long seq_num = 0;
    long t_last_read = 0;
    long av_index = 0;


    //methods
    void init( int dout, int clk){
        scale.begin( dout, clk );
        scale.set_scale( scale_gain );
        scale.set_offset( scale_offset );
        scale.tare(); // Reset the scale to 0

        scale.wait_ready( 1000 );
    };

};
```

Figure 3.1: Arduino Software HX711-Class.

pins and the mentioned init() function. The list is used as a memory structure, because it is easy to iterate through the list and you can access its elements through the index. The Loop Function iterates in each of its iterations though the sensor list. First it is checked in the "Loop function()" whether of the certain functionalities must be carried out. These are explained in the following. In addition then the values of each sensor are read out and output. Last but not least, commands would be processed. The commands would have one parameter more, which describes the list index and thus the sensor to be addressed. Thus the commands could be executed as before. However, all commands are now executed by the host computer. The Arduino Nano no longer accepts commands. The integration of several sensors in the code is thus successful with the introduction of a list and the class "HX711_sensor".

3.2.3 The Software Additions

In the last chapter 3.2.2 all necessary adjustments were described, which took place, in order to take over the original functionality and structure of the software from the balance for this bachelor thesis. However, the sensory book still needs to perform additional tasks. For the manipulation tasks, the internal Inertial Measurement Unit still needs to be addressed. In addition, a WiFi connection is to be established and data sent to the host Computer. The software of the host Computer will then visualize and evaluate the data. Now the software additions for the integration of the Inertial Measurement Unit (IMU) and the WiFi are explained.
The IMU is used to track the orientation of the book. To be able to address the IMU, the "LSM6DS3" library provided by Arduino is used [ard21c] [ard21b]. The name "LSM6DS3" stands for the LSM6DS3 IMU module of the same name, which the Arduino Nano 33 IoT has. In our case the x, y and z coordinates of the accelerometer and gyroscope are read. For this purpose field variables are set. In the "setup()" function the IMU unit is initialized on the Arduino and the sample rates in Hz is output to the serial monitor. In the loop the IMU values are always evaluated first. For this a function "imu()" was created. In this function no more than the values are read and sent to the host computer. The raw values are send - no added calculation on the arduino is needed. The values must be converted on the host computer into the desired unit after receiving. Optionally, if the flag "Debugging" is set to true, the values can still be written to the serial monitor. The values are converted to degrees and the tilt direction is displayed.
The sensory book is supposed to communicate with the host computer via the WLAN and send the data. To do this, the Arduino must first connect to the WLAN. Fortunately, the Arduino Nano 33 IoT already has a WiFi module as the integrated NINA-W102 module. To address this module software-wise, Arduino provides the "WiFiNINA" library [ard21a] **??**. With the help of this library the Arduino will connect to the desired WLAN. In the setup() function the WLAN connection is established as shown again in 3.2.

```
//WIFI
// set the LED as output
pinMode(LED_BUILTIN, OUTPUT);
// attempt to connect to Wi-Fi network:
while (status != WL_CONNECTED) {
  Serial.print("Attempting to connect to network: ");
  Serial.println(ssid);
  // Connect to WPA/WPA2 network:
  status = WiFi.begin(ssid, pass);

  // wait 10 seconds for connection:
  delay(10000);
}
```

Figure 3.2: Arduino Software WLAN Connecting.

The LED lamp will provide information about the WLAN connection at any time and thus provide feedback. Before the setup, the LED lamp present on the Arduino is set off here. In the setup function, a while loop will try to connect to the WLAN through a "WiFi" instance of the WiFiNINA library. Only when the connection is established is the program continued. To connect, the WiFi.begin() function requires the SSID "Service Set Identifier" and the WPA "WiFi Protected Access" password of the available WLAN. These are specified in the fields. With Arduino you have the possibility to manage this data more secure with a so called Secret Tag. The data is written to an extra file, which is read by the main program. The WiFi-begin() function returns a status which is used as a condition of the loop. When the WLAN connection is established and thus the status "WL_CONNECTED" is reached, the loop is finally exited and the program is continued.

```
18:24:24.453 -> Board Information:
18:24:24.453 -> IP Address:
18:24:24.453 -> XXX.XXX.XXX.XXX
18:24:24.453 -> Network Information:
18:24:24.453 -> SSID: WG-Internet
18:24:24.453 -> signal strength (RSSI):-68
18:24:24.453 -> ------------------------------------
```

Figure 3.3: Arduino Software wlan serial monitor .

At the end of setup() the WLAN connection is checked using the created wifi() function. This is also always executed at the beginning of a loop() iteration. On the one hand, the function outputs the connection information. An example for the output iis shown in figure 3.3. Here the IP address of the Arduino, the SSID of the network as well as the RSSI "Received Signal Strength Indicator" is displayed. The RSSI is only a relative value, but the higher it is, the stronger the connection is. However, this output is only optional and is usually not output in the Loop(). But the function also changes the state of the LED lamp so that it blinks. The LED lamp stops blinking when the connection is cut. This provides visual feedback when working with the sensory book.
The last thing to implement is the message exchange between host computer and Arduino via the WLAN. It was decided against a server-client architecture via http. The "User Datagram Protocol" UDP is a communication protocol to send and receive short data packets. To implement this protocol Arduino has provided the WiFiUdp library which is used for this purpose **??**. With the help of the library a WiFiUDP object is created and the port is specified, over which the Arduino can be addressed. In the Setup() function, these aspects are used to initialize the library with the selected port. At the end of the loop function commands are read. In addition to reading the serial monitor, UDP packets are now also received here and included as commands. For this, the original software part that defines commands has been moved to a separate function. The function accepts the command as a string and processes it as before. Thus, inputs can be read in the loop as before and these, then passed to the function, can be executed as

commands. In addition, when a UDP packet is received, it can be read and sent to the read_command(packetBuffer) function for evaluation.

```
void send_data(String msg) {
  int x = Udp.beginPacket("  XXX.XXX.XXX.X ", 12345);
    if(msg == "") {
      Udp.write(ReplyBuffer);
    }
    else {

      Udp.write(msg.c_str());

    }

    Udp.endPacket();
}
```

Figure 3.4: Arduino Software udp packet sending .

Messages are also sent as UDP packets to the host computer. For this the function send_data() was created, which takes the message as string as parameter. The basic concept can once again be found in arduino's documentation [ard21d]. The function is shown again in figure 3.4. A UDP packet is started whith setting address to the host computer with its IP address and the desired port. If the message string is empty, only the debug message is sent that writes that a packet has arrived. This is important for feedback purposes. If a real message is to be sent, it is written to the UDP packet before the packet is closed and sent. The implementation of this UDP packet sending is very simple. The function is always executed after sensor values are read. In more detail send_data() is executed in the functions imu() as well as in hx711(). In hx711() the measurement value is sent after reading each value. The host computer then processes the measurements.

3.3 Host Computer Software

The sensory book is controlled via the host computer. Data is received and processed. Here the idea, the way to the realization and the implementation of the needed running software of the host computer is explained. The software is written in Python contains a script. The script can be executed either in any Python IDE like PyCharm. It writes output to the console and opens a window for user input. The input is processed as commands and affects the functions of the script. Also, certain commands can be sent to the Arduino Nano of the sensory book. Data can and are written to files, so that the measured values can also be evaluated afterwards. The recorded data is plotted afterwards with help of the "matplotlib"-libary.

3.3.1 Idea

The Python script works in a similar way as the software of the Arduino: In a continuous loop. after starting the program a seperate thread is opened, which receives messages of the Arduino in a continuous loop. In addition a small dialog window is opened with an input field and an "Ok" button. When the "Ok" button is pressed, the input is read and processed in the callback() function. When the dialog box is closed, the program also exits.
The data is stored in the fields of the script, but can also be written to a file. Here are the most important fields and their meaning:

- **data_folder**: Here the program folder is specified, in which data can be written and administered. Under certain circumstances, measured values are stored in files here.
- **UDP_IP, UDP_PORT** : Here the IP address of the Arduino and the port of the Arduino are set. These are essential for sending the UDP packets to the Arduino.
- **hx_databankl**: Here the measured values of the Load Cells sent by the Arduino are managed. With a quantity of 50000 the list is emptied automatically, in order to prevent a possible OutOfMemmoryException. A data tuple is structured as follows: [Past milliseconds since the Arduino started, iteration number of the loop function of the Arduino, raw sensor value, offset to calculate the zero point, sensor ID, calculated sensor value in grams].
- **hx_book_databank** : The calculated measured values for the entire book are managed here - as tuples. A tuple consists of four elements which are structured as follows: [total weight on the grid of the book, weight on the platform for book tilting, first coordinate for force-mutual point position determination, second coordinate for force-mutual point position determination].
- **imu_databank**: In this list all measured values of the IMU of the Arduino are managed in the sensory book. The list is also automatically cleared after 50000 entries to prevent out of memory exceptions. The data is also in tuple form here and contains the following values: Past milliseconds since the Arduino started, IMU-X value, IMU-Y value, IMU-Y value]

.

The separate thread executes the function "listening_thread()". This function reads the data in a while-true loop and processes it. The socket library is used to receive the messages and decode the data. Afterwards the message is processed. For this it is checked with which characters the message starts. Messages from the Arduino that start with "acknowledged" are only responses from the Arduino to user requests. These are output

to the console for feedback. This way the user is always aware if the commands have arrived at the Arduino. Then there are two more message types: Messages with the values of the IMU and messages with the values of a Load Cell. If the message starts with "imu", it is a tuple with the IMU values. The values are then appended to the internal IMU data list "imu_databank". In addition it is checked here that the list does not become too large. If there are more than 50000 entries, the list will be cleared automatically.
If the message starts with "W", it is a Load Cell data tuple. These are processed and appended as a tuple to the list of all Load Cell measured values "hx_databank". Thereby the measured value of the sensor is used to calculate the weight on it:

$$Value = (rawValue - offset) \cdot calibrationFactor \cdot regulationFactor$$

The offset is the value that the sensor measures when no weight is placed on the sensor. The calibration factor converts the value into grams. The regulation Factor let all Sensors in the same direction: If weight is put on the book, all sensors should measure values in the positive number range. The sensor data always arrive one after the other. Thus the message with the measured values of the fifth sensor always comes last and closes a loop iteration of the Arduino. At the last iteration value the function "get_real_hx_values" is executed. In it a new entry for the hx_book_databank list is calculated. On the one hand, the total weight acting on the book is calculated from all sensor values - The weights or forces pressing on the book at the bottom can be calculated. If a force acts on the book, the function tries to find the position of the book where the force acts. For this, the sensor values are added together, which are diagonally opposite each other. Then the respective percentage of the summed weight is calculated. From this two coordinates follow, which give information about the position of the force. The function outputs the values as a tuple, which is then appended back into the list hx_book_databank in listening_thread(). Also, if there are more than 50000 items in the list, it will be cleared. Additionally, if the command was entered by the user, the offsets or the calibration factors are recalculated. The respective boolean variable "offset" or "calib" is then True and the code of the respective if-condition is executed. For the calculation of the offset only the current raw value of the measurement is taken. For the calculation of the calibration factor the function calib_hx() is called. Here the total mass of all Load Cell raw values of an Arduino loop iteration() is added together and then it is calculated how big the shares of the Load Cells are. From this the proportional weight is calculated, which lies on the sensor. For this the function needs additionally the real weight of the object, which lies on the book. The value measured by the sensor is divided by the actual weight acting on the sensor and the result is taken as the new calibration factor for the sensor:

$$MeasuredTotalMass = rRawValue1 + rRawValue2 + RawValue3 + RawValue4$$
$$PropotionX = sRawValueX / MeasuredTotalMass$$
$$CalibrationFactorX = (PropotionX * MeasuredTotalMass) / RawValueX$$

In addition, the sensor for tilting the book is calibrated with the same calculation. This includes all functionalities of the thread that is responsible for message reception and data processing.
The main thread waits in a continuous loop for an input from the user, which must be entered into the dialog box and confirmed with "Ok". In this continuous loop the callback() function is executed. If "Ok" was pressed and the input field contains characters, the input is tried to be interpreted as a command. in the following is a list of all commands that can be sent:

- **offset**: If the command starts with "offset", the offset is recalculated
- **calib book int**: The four sensors holding the grid are calibrated. The weight of the object lying on the book is given as int.
- **calib_tilt int**: The sensor that holds the platform to measure the tilting is calibrated. The weight of the object lying on the platform is given as int.
- **start** : Execute command to start recording for a manipulation task. The offsets should be set beforehand and calibration factors calculated.
- **stop FileName**: Stops recording and should be executed after a manipulation task is completed. Parameters: FileName is the desired filename prefix of the to be created files with the recorded data.
- **default**: Any other command is sent to the Arduino Nano in the sensory book. The commands of the Arduino can be found in the previous chapter.

.

This already includes the entire software on the host computer with all its input possibilities. With this software the manipulation tasks will be tracked on the Arduino and the experiments will be performed.

4 Experiments

The first printed book is used for the experiments. A second version could not be printed and processed functionally in time.
The goal is to find out whether the implementation of sensors in the manipulated object has advantages and provides more valuable data than the more common methods. In this chapter the experiments are described and explained once. before an experiment can be started precautions have to be taken. On the one hand the sensory book and the host computer python script have to be restarted. This is to ensure that the experimental conditions are always the same. Then it is checked that the data packets really arrive at the Python script. The console on the host computer provides information on whether the connection is established in both directions and the user can check if message packets can be transmitted from both sockets. Afterwards the offsets for the sensors have to be set, to determine the zero point. For this, the book must be placed quietly on a table with the grid facing upwards. No weight may lie on the book. Then, in the dialog box on the host computer, the offsets are reset with the command "offset". After making sure that the sensors measure values around zero (fluctuations are unfortunately present), the sensors are now calibrated. For this, first an object is placed on the lying book, more precise on the grid. It is important to know the real weight of the object. In the case of the experiment a $345g$ heavy cup was used. After the cup has been moved onto the book, the command "calib_book 345" can be used to calibrate the four sensors of the grid. Afterwards the script gives the weight with small errors due to fluctuations. Last but not least, the book is placed on its side so that the platform for the measurement is on top when the book is tilted. The cup is now placed on this and the sensor is calibrated with the command "calib_tilt 345". Then an experiment can start. The procedures of all experiments are now presented and explained once in succession.

4.1 Pushing a book across a table

For this manipulation task, the book is placed at the edge of the table so that the book grid faces upwards. After recalculating the offsets and calibrating the grid sensors, a try can be started. The recording on the host is started and the book is pushed in one direction at a slow but natural speed. The book is gripped with the hand on the book cover or book grid - just as a book is usually shifted when it is lying down. Visualized in Figure 4.1 are the four attempts in this Experiment:

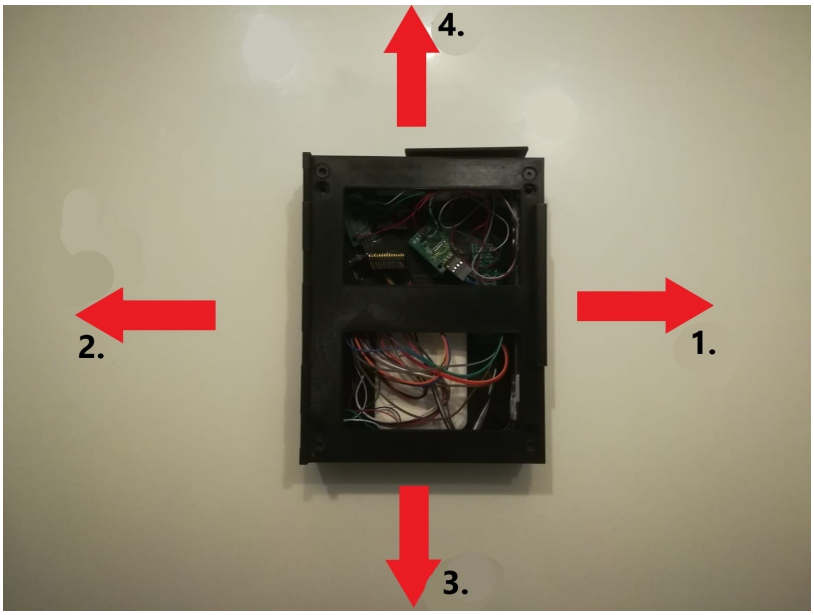

Figure 4.1: Pushing a book across a table.

1. The book is placed at the left edge of the table and moved to the right.
2. The book is placed at the right edge of the table and moved to the left.
3. The book is placed at the top edge of the table and moved down to the bottom edge.
4. The book is placed at the bottom edge of the table and moved up to the top edge.

After the task is completed, the recording is stopped. In the evaluation the recorded data will be used.

4.2 Picking up a book from a table

In this manipulation task, the book is also placed on the edge of the table. Since the fifth sensor does not work with this book, the experiment must be implemented differently. The book slightly overhangs the table, so that the book can be grasped at the bottom and top and thus lifted. This is visualized again in Figure 4.2. After adjusting the offset and calibration variables, the experiment is also started here with the dialog entry "start". After the book has been lifted and then placed back on the table, the test is ended with the command "stop fileName". The data is then analyzed. The experiment is repeated twice to reduce errors or noise.

4.3 Re-grabbing a book with both hands

The experimental setup is very similar to the second experiment. In this case, however, the book is gripped with both hands and lifted into the air. Here we measure whether

Figure 4.2: Picking up a book from a table

there is a visible difference to the second experiment.

4.4 Missing Experiments

Originally, two more tests were planned, but unfortunately they could not be carried out:

1. Pulling a book from a shelf between other books.
2. the book and turning a page.

The former was not possible because the responsible sensor does not output any values. Due to the current circumstances many steps took longer and so a second version of the book could not be printed and made functional fast enough to be working with. In the second book the deficiencies should be eliminated, also the sensor could be exchanged or corrected. Further work on the book could generate more insight - the experiments that are possible through the broken sensor promise many insights. As the book is available for experimentation, only the two larger surfaces are capable of absorbing forces, and experiments must engage with them. Nevertheless, conclusions about the quality of the data can be drawn from the experiments.
The last originally intended experiment could not be carried out either. The mechanics of how to measure the turning of a page is missing.

5 Evaluation

The design, the software, the sensory book in its first real form as well as the results of the experiments are evaluated and critically examined here. The data of the experiments are analyzed and evaluated. Conclusions are drawn about the question whether sensors in the manipulated object are worthwhile.

5.1 Design Evaluation

In design you try to make as few mistakes as possible. The final product should be as good as expected - just like in a software project. In this case, however, the entire development is carried out without intermediate results. The final result is printed and the first errors become visible. Thus, the first book that was printed in its entirety has many imperfections. As the largest deficiency first the drillings do not agree with the used sensors. It was immediately learned that every little thing has to be checked: The sensor models from the paper [**?**] were taken over. Since length, width as well as bore sizes corresponded, it was assumed that the model corresponds to the real sensor. In retrospect it could be determined that with the real sensors the distances between the two drillings of a side are larger than the model ones in each case around $5mm$. Thus, the sensors could only be attached to the book through one of the drilled holes per side.
Furthermore, the book grid is not stable enough and can bend if too much force is applied. Only in the second book the grid can gain stability with thicker struts. A big problem is the mechanics chosen for the cover page. The idea of clipping the page onto the grid could not be made true. The material is harder than expected. If the hinges on the cover are too thin, they break. If the hinges are too thick, the present problem occurs: It can't be connected. Another mechanism would have to be used that does not require such clipping, or the appropriate dimensions would have to be found. Also, the distance and thus the gap between the grid and the book can be reduced - a gap of only $1mm$ is enough. Larger inconveniences still became clear during the placement. First, the Ardiono must be placed upside down on the surface. The pins of the Arduino Nano would be clearly better attainable in that way. Furthermore, the distance between the Arduino and the Power Bank is very narrow. You can connect the components with a cable, but the cable must be very curved or form a big curve inside the book. For the first book usually long cables were used, which makes the inside very messy
. As a last critical point, the choice of a grid is critically considered at the end. It makes the sensory object lose too much of its book-like appearance and the surface can no longer be

fully used - the cover page should never measure forces.
There is a lot of room for improvement and especially the book page mechanics should be revised. Anf one big aspect is missing: The turning of the first pages cannot be captured.

5.2 Software Evaluation

The software of the Arduino works in the aspect that all measured values are read and sent. It connects to the WLAN without problems and the messages arrive at the host computer. The problem is still the time: the values are sent only every 1.3s. Surely there is a way to increase the frequency of the measurement data and thus capture faster movements more accurately. However, with such a slow frequency, normal movements cannot be accurately reproduced. As a consequence, the experiments have to be performed in slower movements. This must have the consequence that the experiments were distorted and real sequences could not be reproduced. Also, the Arduino Nano takes several minutes to boot up and take the first measurements. This is not an experiment-essential shortcoming, but it is an inconvenience. It takes several minutes to see if the system has booted up properly. A small system of this type should not need such a long startup process. During experimenting it was found out that probably loose contacts slow down the book. In some cases Arduino loop frequencies of 0.2s per loop could be achieved. However, this could not be maintained due to the wiggle contacts.
The Python script on the host computer also serves its purpose. The system could be equipped with a better User Interface (UI) for further work. The makeshift UI with a simple dialog box is sufficient for initial experimentation, but should be tweaked with more time.

5.3 Data Evaluation

Due to the delays and consequent lack of time, the sensory book of the first version had to be used for the experiments. The experiments were not carried out in the university, which meant that cameras were not used to track the book.

5.3.1 Sensor Values

It is noticeable that the fifth sensor does not provide any data - it always provides the same value. Then each sensor has fluctuations - whether at idle or with weight on the sensor. These fluctuations are particularly large for sensor "3" (top left). This hinders the setting of the offsets to reach a zero point. The values of the total mass on the book are between -2000 to + 8000 in the not calibrated state - These large fluctuations are especially due to the third sensor. But even so, there are fluctuations in general. After calibration with the 345g heavy cup, a value of 335 - 345 is measured. The values can therefore

be less than 5% away from the real value only falsify the result in small dimensions. In addition, the data arrives only every 1.3s. The frequency of the data is catastrophic. The experiments must be approached with aggravated conditions and do not promise anything good.

5.3.2 Experiment 1 - Pushing a book across a table

The data for the experiment "Pushing a book across a table" are unfortunately hardly good to evaluate. Due to the poor frequency of the data, only about 10 to 14 data sets are available for one experiment. The frequency is too poor to draw plausible conclusions. The resulting plots for the weight acting on the book all have a similar shape. The plots for experiments 1.1 and 1.2 are shown here once in Figure 5.1 and 5.2. It can be seen that the book was moved for only a few seconds in each case. In Figure 5.1 you can see that the book was moved between time-step $2-6$. Due to the long time jumps between the time-steps, a detection of acceleration or weight is hardly recognizable. The graph rises abruptly steeply, then holds at over $400g$ and finally drops again just as abruptly at time-step 7. The force can be calculated by $F = m \cdot a$. This results in an average value of $0.466kg \cdot 9.81m/s^2 = 4.57N$ in the first try and $0.998kg \cdot 9.81m/s^2 = 9.79N$ in the second try. It is also easy to see here that the true zero point is never reached. Since the sensors have measurement fluctuations, there is always an inaccuracy of -5% to $+25\%$ in the result.

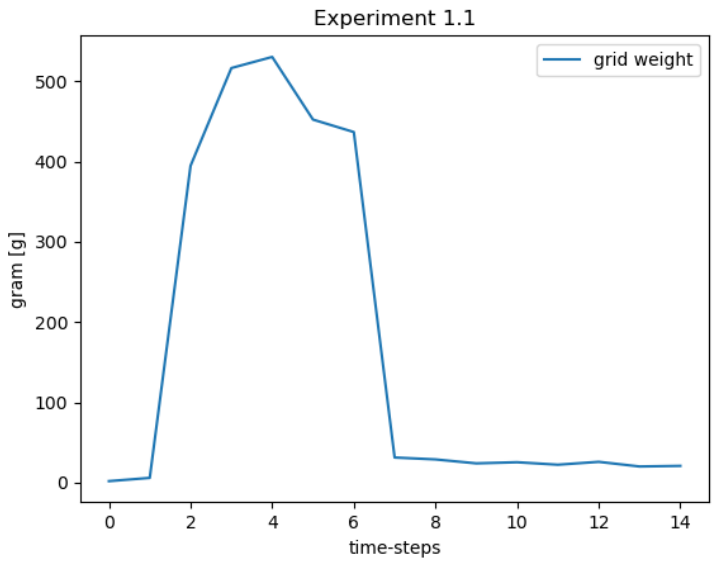

Figure 5.1: Experiment 1-1 Book-Weight over Time .

This also falsifies the position measurements. Especially in the steady state, the values tear out very much because of this. The figures 5.3 and 5.4 show the position values after the time. The values outside the movement time span should not be used for analysing - Unfortunately, the sensor fluctuations, which also go into the negative range, completely

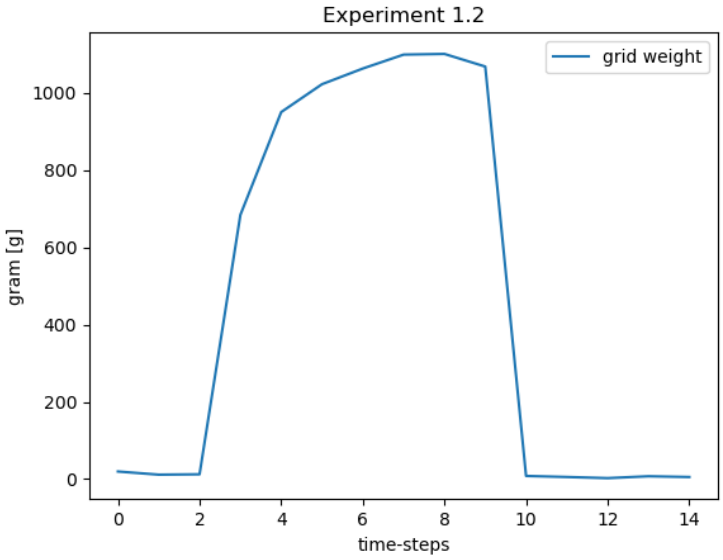

Figure 5.2: Experiment 1-2 Book-Weight over Time .

destroy the readability of the plots. Therefore for the first attempt "left to right" of the experiment here again the values are listed in table 5.1.

Timestep	posX*	posY*
0	-54.38	118.90
1	52.02	-10.21
2	68.35	252.231
3	68.49	247.95
4	72.31	241.35
5	76.01	250.74
6	74.97	506.21
7	92.83	552.50
8	92.25	123.11
9	91.14	268.43
10	91.14	129.93
11	91.53	8028.57

Table 5.1: Experiment 1.1 Position x* and y* over time

The values are to be understood in the table 5.1 as follows: The two values are direction-vectors, each pointing diagonally opposite each other from one book corner to another book corner, as further illustrated in Figure 5.5. The values are given in millimeters. For the meaning of the values follows an example for the point of contact at time step 4: If one chooses the center of the book, where both vectors intersect, as starting point of the orientation and holds his finger up there. Then the finger must be moved in direction x* by $72.31mm - (192^2mm + 152^2mm)/2 = -50.17mm$. Afterwards the finger is shifted by $241.36mm - (192^2mm + 152^2mm)/2 = 118.92mm$ in direction y* and the calculated value

is found. This results in a value in the range of the red area drawn in the figure 5.5 - probably even a little bit higher. The exact value is outside the book. This must be a result of the fluctuations as well as imperfect calibration factors. However, if you increase the radius of the point so that it hits the book, you capture approximately the exact area that was really touched by the hand in the experiment. Of course, the point only indicates the center of gravity of the force application, since nothing is known about the area of the hand. So the position detection is not perfect, but it works with some adjustment. These adjustments could be solved by software for further work.

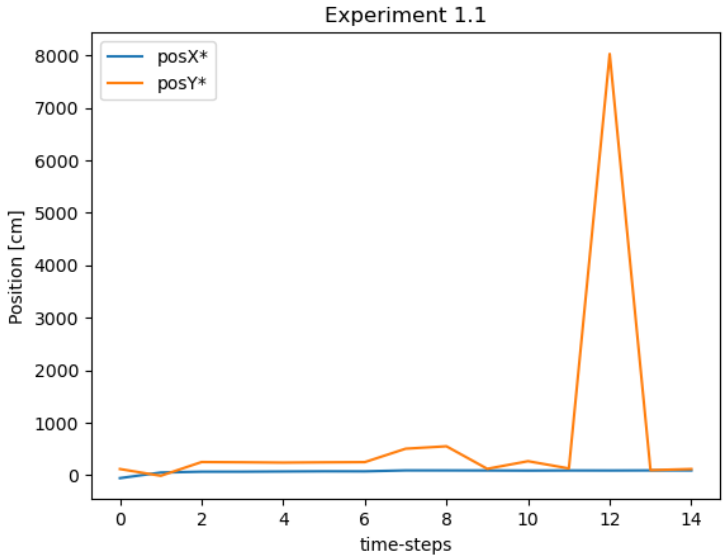

Figure 5.3: Experiment 1-1 Position x* and y* over Time .

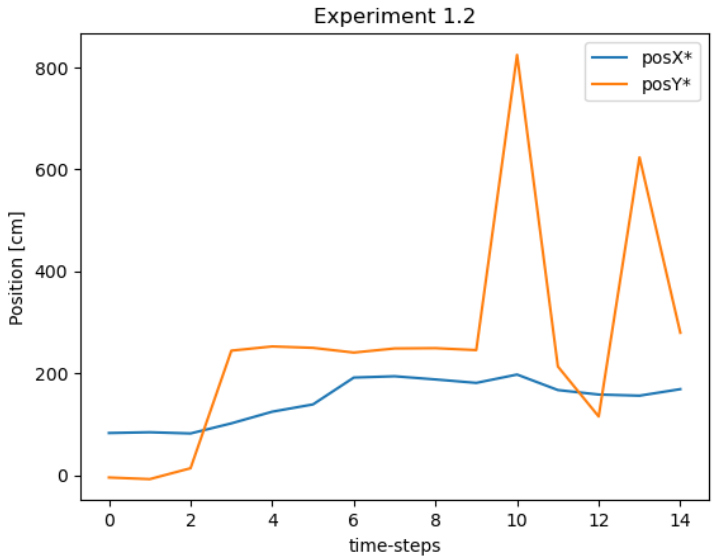

Figure 5.4: Experiment 1-2 Position x* and y* over Time .

Figure 5.5: Experiment 1-1 Position on Book

The values of the IMU do not give any information. The orientation has not changed. Since the Arduino in the used book is only stuck with scotch tape and is therefore not in perfect position, the gyroscopes and accelerators x-y-z coordinates do not agree with those of the book and overlap very much. In addition, no inference can be drawn from the values because the frequency at which they occur is so low. The Y values of the accelerometer have changed from -2.5 in steady state to -4 or even $+5$ in moving state. However, the jumps are not traceable and one cannot form any conclusions from them. On the whole, one must interpret the experiment as a failure. The small amount of data allows few conclusions. A force has been recorded and also the position finding of the force has succeeded with a certain interpretation effort. But one cannot put it in relation to the event. There are no camera images or other possibilities to see exactly the speed or acceleration of the book. However, the force $F = m \cdot a$ can be calculated. However, since the book was pushed slower than usual because of the low frequency rate, this value cannot be chosen as realistic.

5.3.3 Experiment 2 - Picking up a book from a table

In this experiment, the values are also in such a weak frequency that they cannot really be evaluated. The experiment was repeated three times, but the data on each try speak the same story. Figures **??** and **??** plot the applied weight on the book over the course of each experiment. As in the previous experiment, the data are not smooth because the time jump of the data points is $1.3s$. Also, it can be seen that the weights are plotted in the positive range. As it appears, the attempt to reduce incoming forces acting on the bottom of the Book could not be represented in the negative number range.

In Figure **??**, as in the first experiment, there is a strong abrupt increase as well as

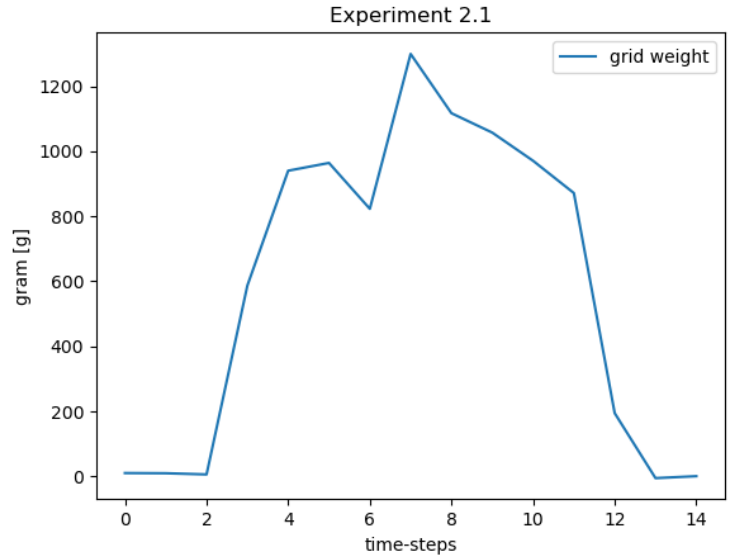

Figure 5.6: Experiment 2-1 Book-Weight over Time .

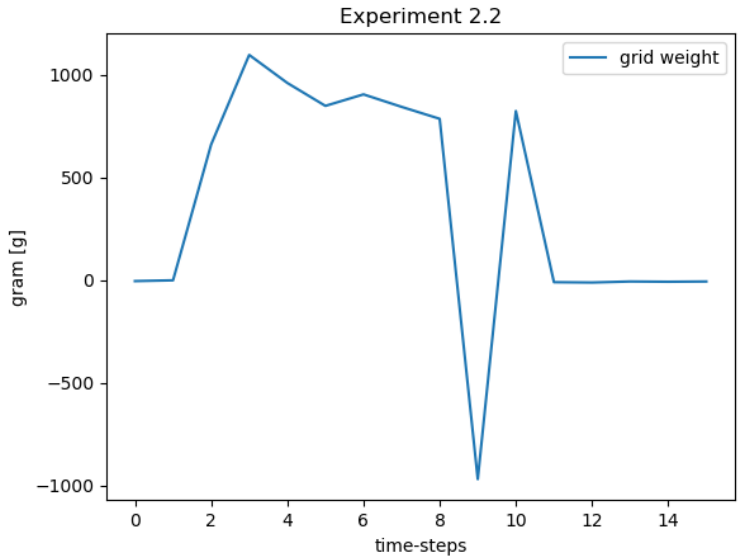

Figure 5.7: Experiment 2-2 Book-Weight over Time .

decrease. However, the fluctuations during the movement are striking - the book received different force impressions while it was swung in the air. But because of the bad data frequency only a strong rise and descent can really be interpreted. One cannot draw any other conclusions. In figure **??** it does not look better. But there a sensor must have measured an error value in time-step 9, which set the calculated value to $-1000g$. Except for this case, one sees basically the same picture as in the other experiments: An initial sharp increase. Then the value holds above a certain threshold and finally decreases rapidly.

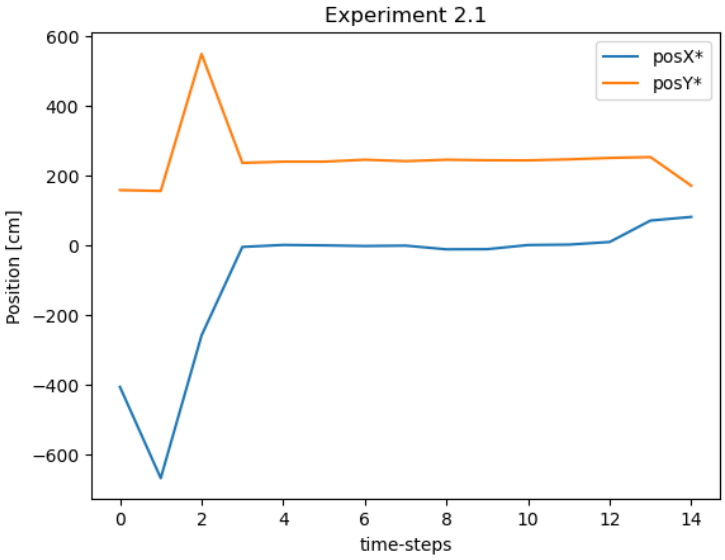

Figure 5.8: Experiment 2-1 Position x* and y* over Time .

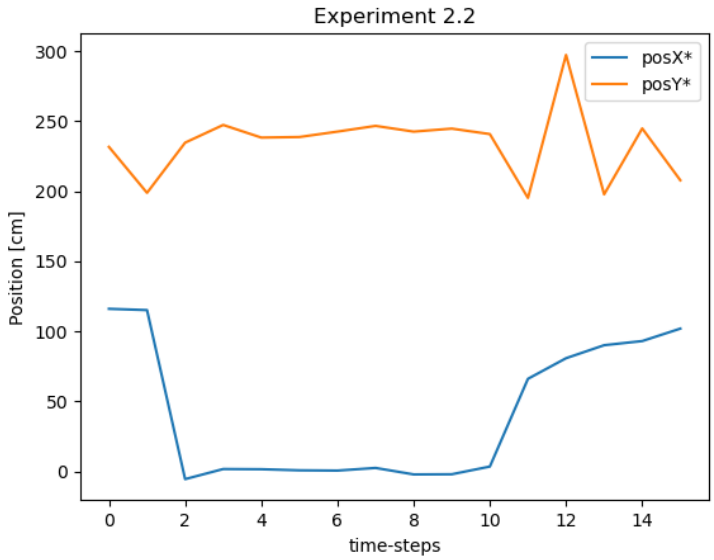

Figure 5.9: Experiment 2-2 Position x* and y* over Time .

The values of the position determination for this experiment do not match the real touch points. In figures 5.8 and **??** one can see similar values in the moving area. In figure 5.8 between time-step 2 to 10 the value of x* is about 0mm and y* is about 240mm. This results in a position determination using the same method as for the dark red cross in figure 5.10. The light red cross marks the real calculated position, the green cross the real point of contact. The position determination was not correct in this experiment, the position coordinates are similar to the first experiment, but not the same. Therefore it can be assumed that there is no software technical error, which always outputs the same

value.

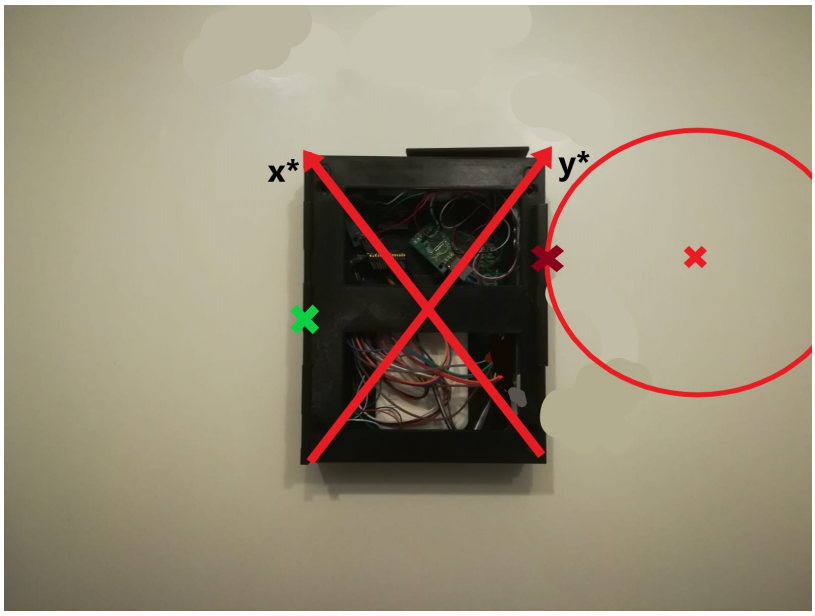

Figure 5.10: Experiment 2 Position on Book

As in the first experiment, the IMU values give little information. The book was changed in an orientation and changed data are evident, but the weak frequency does not allow any conclusions. Again, the experiment is hampered by circumstances in the private home. The missing cameras do not allow any conclusions about speed and therefore there is little relation to reality.

Also this experiment failed, insofar that no answer to the question occurred, whether sensors in the manipulated object yield a lower bias or error rate. On the contrary, one would have to assume from this that the sensors in the object deliver worse data - In the case of this work, however, the error lies in the unfinished implementation and worse experimental conditions. Once again it becomes clear that the first version of the book does not work properly.

5.3.4 Experiment 3 - Re-grabbing a book with both hands

This experiment gave interesting conclusions about the book. The values were exactly the same as those of the second experiment and no difference could be found. The determination of the position of the force application is not of great importance here, since there is more than one point of contact. Nevertheless, the position determination worked relatively well, as can be seen in Figure 5.11. The true center of gravity of the acting forces had to lie approximately at the green cross. The calculated area lies in the red circled area. The positioning here is not perfect and has large fluctuations - but recognizes the approximate direction.

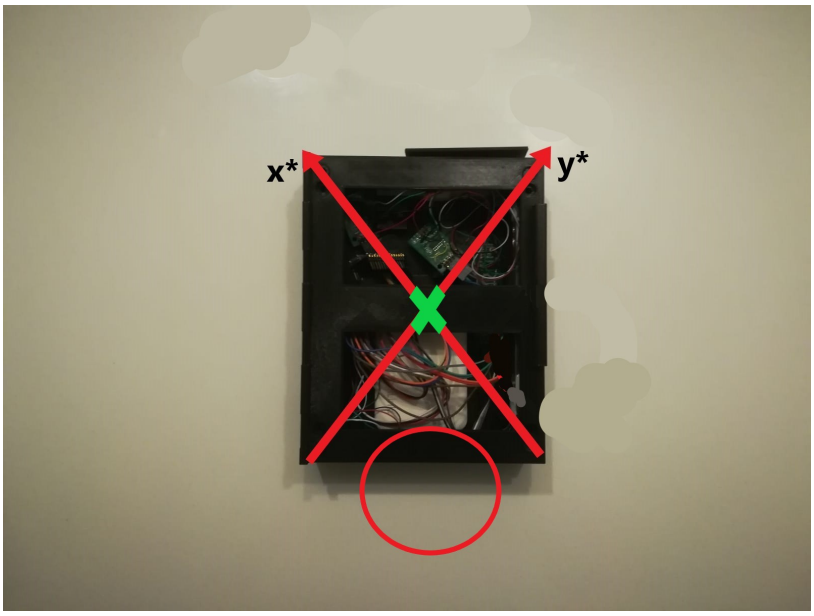

Figure 5.11: Experiment 3 Position on Book

But with this experiment something else was noticed: There must be wiggle contacts in the circuit. At the second run suddenly the data frequency jumped from 1.2*s* to 0.2*s*. This is a much more attractive frequency to work with scientifically. However, the frequency was not of long duration and fluctuated relatively frequently and finally it fell back to the frequency of 1.2*s* on several restarts. This again gives the hint that the book could be adapted and reprinted, rewired and provided with new hardware parts, probably with a much better data material.
Unfortunately, the experiments could not be repeated again with increased frequency because the frequency could not be increased again. The third experiment was concluded with the same conclusion as the second: no conclusions can be drawn from the data that have not already been made in the first experiment.

6 Conclusion

Robotics remains a very interesting field. A field in which even the smallest things can cause problems. The sensory book does not work to the extent that you could even begin to compare its measurements with those of a professional sensory glove. Sensory gloves have found a great deal of influence in research and, as in Paper **??** or Paper **??**, research continues into them and new uses. It lacks in too many corners for that. That a glove does not deliver perfect values is undoubtedly true - but whether a lower error rate can be achieved if the sensors are built into the manipulated object remains open. Sensory gloves remain widely used in the research world The assumption stays, however: If the object can measure the values without problems, if the object can be in the same dimensions, weight and appearance as the original, so that it feels and can be manipulated exactly like the original, then there should be better results. A glove must have some negative effect. This is also mentioned and tried to be addressed in the paper [AMOL].
A certain error rate is present - but it is usually not significant. Some inaccuracy is mentioned here, but it is perceived as not significant. How much influence the reduced sense of touch has in the inaccuracy is not clarified here. The motivation to research into a sensory object remains.
The sensory book is a perfect example of how, in computer science and especially robotics, even the smallest and simplest seeming aspects can complicate a project. The smallest variations in sensors, the smallest decisions about design, or the wrong lines of code can significantly drag out a project. The complexity of a problem should never be underestimated.

Bibliography

[acc21a] Alles über Accelerometer und Gyrometer. (seen 2021). https://de.rs-online.com/web/generalDisplay.html?id=ideen-und-tipps/accelerometer-leitfaden

[acc21b] Definitions. (seen 2021). https://gladiatortechnologies.com/glossary/?gclid=CjwKCAiAm7OMBhAQEiwArvGi3GccEUHqp5_GtaoQaFzLpj4xy1-gco4akKtkjccKWW7oluMmnTvvfBoCeB8QAvD_BwE

[AMOL] ALLEN, Peter K. ; MILLER, Andrew T. ; OH, Paul Y. ; LEIBOWITZ, Brian S.: Integration of Vision, Force and Tactile Sensing for Grasping.

[ard21a] Connecting the Nano 33 IoT to a Wi-Fi Network. (2021). https://docs.arduino.cc/tutorials/nano-33-iot/WiFi_connection

[ard21b] Accessing Accelerometer Data on Nano 33 IoT. (seen 11.2021). https://docs.arduino.cc/tutorials/nano-33-iot/imu_accelerometer

[ard21c] Accessing Gyroscope Data on Nano 33 IoT. (seen 11.2021). https://docs.arduino.cc/tutorials/nano-33-iot/imu_gyroscope

[ard21d] Send and Receive UDP String. (seen 2021). https://www.arduino.cc/en/Tutorial/LibraryExamples/WiFiNINAWiFiUdpSendReceiveString

[boo21] BUCHFORMATE UND ABMESSUNGEN. (seen 2021). https://www.blurb.de/book-dimensionsl

[car20] Robotics and automation in automotive manufacturing and supply chains. (2020). https://roboticsandautomationnews.com/2020/02/24/special-report-robotics-and-automation-in-automotive-manufacturing-ar30374/

[HWZ20] HENDRICH, Norman ; WASSERFALL, Florens ; ZHANG, Jianwei: 3D Printed Low-Cost Force-Torque Sensors. (2020)

[Kha19] KHALIQ, Siraj: Wie die Robotik den Weg für die Industrie 4.0 ebnet. (22.02.2019). https://www.maschinenmarkt.vogel.de/wie-die-robotik-den-weg-fuer-die-industrie-40-ebnet-a-778196/1

[loa21] Strain Gauge Load Cell - 4 Wires - 5Kg - 80mm long. (seen 2021). https://www.adafruit.com/product/5230

[nan21a] *Arduino Nano 33 IOT*. https://www.distrelec.de/de/arduino-nano-33-iot-arduino-abx00027/p/30150883. Version: 2021

[nan21b] Arduino Nano technical specs. (seen 2021). https://store-usa.arduino.cc/products/arduino-nano-33-iot

[OH21] OWEN-HILL, Alex: Top 6 Robotics Industries That Are Growing in 2021. (2021). https://robodk.com/blog/top-6-robotics-industries-growth-2021/

[pb-21] LOGILINK PA0207 Power bank, Li-Po, 3000 mAh, USB. (seen 2021). https://www.reichelt.com/de/en/power-bank-li-po-3000-mah-usb-logilink-pa0207-p270561.html

[RLM21] ROSENCRANCE, Linda ; LAWTON, George ; MOOZAKIS, Chuck: User Datagram Protocol (UDP). (seen 2021). https://www.techtarget.com/searchnetworking/definition/UDP-User-Datagram-Protocol

[str21] Strain Gauge Load Cell Basics. (seen 2021). https://www.800loadcel.com/load-cell-and-strain-gauge-basics.html

[udp21] UDP - User Datagram Protocol. (seen 2021). https://www.elektronik-kompendium.de/sites/net/0812281.htm

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.
Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den _____ Unterschrift: _____